开源安全运维平台

Open Source Security Operation and Maintenance Platform

OSSIM Best Practices

OSSIM最佳实践

李晨光 著

Big Data

Open Source
Security Information Management

Cloud Computing

Internet of Things

清华大学出版社

开源安全运维平台

OSSIM最佳实践

李晨光 著

清华大学出版社
北 京

内 容 简 介

在传统的异构网络环境中，运维人员往往利用各种复杂的监管工具来管理网络，由于缺乏一种集成安全运维平台，当遇到故障时总是处于被动“救火”状态，如何将资产管理、流量监控、漏洞管理、入侵监测、合规管理等重要环节，通过开源软件集成到统一的平台中，以实现安全事件关联分析，可从本书介绍的 OSSIM 平台中找到答案。本书借助作者在 OSSIM 领域长达 10 年开发应用实践经验之上，以大量生动实例阐述了基于插件收集日志并实现标准化，安全事件规范化分类，关联分析的精髓，书中为读者展示的所有知识和实例均来自大型企业中复杂的生产环境，并针对各种难题给出解决方案。

全书共分三篇，10 章：第一篇（第 1~2 章）主要介绍 OSSIM 架构与工作原理、系统规划、实施关键要素和过滤分析 SIEM 事件的要领。第二篇（第 3~6 章）主要介绍 OSSIM 所涉及的几个后台数据库，重点强调安全事件分类聚合、提取流程、关联分析算法、Snort 规则分析等技巧。第三篇（第 7~10 章）主要介绍日志收集方法和标准化实现思路以及在 OSSIM 中用 HIDS/NIDS、NetFlow 抓包分析异常流量的方法，深入分析了 OpenVAS 架构和脚本分析方法。

本书可以作为开源安全技术研究人员、网络安全管理人员以及高校计算机专业师生学习参考使用。

图书在版编目（CIP）数据

开源安全运维平台：OSSIM 最佳实践 / 李晨光著. - 北京：清华大学出版社，2016
ISBN 978-7-302-42385-0

I. ①开… II. ①李… III. ①Linux 操作系统－安全技术 IV. ①TP316.89

中国版本图书馆 CIP 数据核字(2015)第 296359 号

责任编辑：夏非彼
封面设计：王　翔
责任校对：闫秀华
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市中晟雅豪印务有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张：**42.5　　**字　　数：**1088 千字
附光盘 1 张
版　　次：2016 年 1 月第 1 版　　**印　　次：**2016 年 1 月第 1 次印刷
印　　数：1~3000
定　　价：148.00 元

产品编号：062466-01

媒体推荐

51CTO 推荐

认识晨光近十年，旁观了他网络管理实践水平的日臻高超，也见证了他多年来在知识及技能的整理与传播方面的坚持和硕果累累。晨光的文章和著作特点明显：结合实践、平实厚重、干货多多，也因此受到读者的爱戴。在本书发表之前，晨光已经发表了 OSSIM 的博文六十余篇，可想而知他为本书出版的积累之深。我相信，阅读此书，您一定收获满满！

杨文飞　51CTO 总编

51CTO 推荐

李晨光老师是 51CTO 专家博主，也是 51CTO 学院知名讲师。他的文章深受同行关注和认可，荣获了多项殊荣，他的课程在学院深受学员喜爱。新书《开源安全运维平台——OSSIM 最佳实践》是李老师在 OSSIM 领域长达 10 年开发应用实战经验的总结和凝练，书中的所有知识和实例均来自大型企业中复杂的生产环境，并针对各种难题给出解决方案，相信此书一定会深受广大读者的支持。

51CTO 社区、51CTO 学院

CSDN 推荐

说来惭愧，身为机房装机工出身，竟然也是 Google 了半天才搞清楚什么叫 OSSIM。当年在机房苦哈哈地安装、调测 Nagios 和 Snort 的场景还历历在目，读起晨光老师的这本《开源安全运维平台——OSSIM 最佳实践》自然倍感亲切。

我的理解，OSSIM 是安全运维发展到一定阶段后体系化、工程化的成果，强调独立安全应用间的配合。这对使用者提出了很高的要求：不仅要熟悉系统中每个应用的用法，还要清楚安全信息在整个系统中的流转，以及出了问题后的准确定位。这不仅需要大量的实践，更需要经验的积累。这本书的内容，能更快地帮助你了解这个复杂的系统。更为难得的是，书

中包含了很多最佳实践的分享，对于有一定经验的读者也有着很好的参考意义。

作者晨光老师在安全运维领域耕耘多年，书中的内容，都是在常年工作中总结出的实战经验之谈，是国内第一本系统阐述 OSSIM 理论和实践的作品，更是一个有追求的运维人员在成长路程上不可多得的资料。

然而让人觉得非常可惜的是，国内系统化安全运维的理念并不普及，提起 OSSIM 来，估计知其然的人就不多，更勿论知其所以然者。希望晨光老师的这本书能被更多的人了解和学习，能切切实实地帮到奋斗在一线的兄弟们。我想，这也是每一个 CSDN 人的愿望。

李申 CSDN 社区运营总监、CSDN 学院总监

IT168 推荐

多年来，李晨光老师一直是众多 IT 圈朋友的良师益友。文章素来结构清晰，布局平实，技术内容扎实，读后受益良多。欣闻李晨光老师将发新作，想来又是一次技术升华的历练旅程。在作者的描述下，OSSIM 这一还尚处于发端的全新安全运维架构扑面而来，读来不忍释手。原因有三：

其一，信息安全市场历来与开源没有渊源，不管是 SIEM 还是 SOC，在国内普及和实践也有些时日，但一直不温不火，搭上开源是偶然还是必然，以求甚解；其二，诚如作者所言，“本书不是神功秘籍”，只为读者铺陈经验、答疑解惑。而信息安全之于企业是个平衡问题，安全运维如何在快速变化的动态中，找到最佳平衡支撑，恰恰需要兼容并蓄；其三，想来，数据驱动的浪潮或已不远，安全威胁的全息生态掌控，玩的正是数据、事件和风险的收集和分析，要主动出击还是被动防御，不难选择。作为先睹为快者，我只能谈一些浅见，是为序。

陈毅东 IT168 企业级副总编

启明星辰专家推荐

安全信息与事件分析（SIEM）技术进入中国也有十几年了，但是就如同以 SIEM 为核心的安全运营中心（SOC）一样，由于顶着过于炫目的光环，在国内的发展始终喜忧并存。究其缘由，其中很重要的一点就在于 SIEM 是安全分析的集大成技术，涉及面广、复杂度高，对使用者要求也比较高，而国内信息安全产业的发展以及安全运维体系还未完全成熟。

但是，安全事件分析作为安全运维的核心技术无可替代，是企业和组织信息安全建设以

及安全运维的必然选择。在这种背景下，国内迫切需要一系列的相关书籍来传播和推广相关技术，本书无疑是国内安全事件分析技术领域的重要论著。

从全球范围来看，SIEM 技术发展已趋于成熟，2014 年市场规模接近 17 亿美元，商业公司占据了大部分的市场。其中，OSSIM 是唯一成功的开源 SIEM。从 2003 年发布第一个版本至今，OSSIM 已经发展了 12 年，足见其强大的生命力。而以 OSSIM 为基础成立的 AlienVault 公司也已经成为 SIEM 领域的知名公司。

OSSIM 作为一个开源安全运维和安全事件分析平台，较好地集成了各种开源的安全工具，并能够与大量商业化的安全产品进行对接，同时还具备很强的扩展能力，真正成为一个安全运维的开放式平台。

李晨光先生是国内 OSSIM 领域的权威人士，对安全运维有深刻的理解，拥有丰富的实战经验，书中汇集了他多年的实践成果，十分难得。跟随晨光学习 OSSIM，不仅能够提升自身的安全运维实战能力，也有助于理解安全运维体系和安全事件分析运作原理。

叶蓬 启明星辰泰合 SOC 产品总监、SOC 布道师

ChinaUnix 推荐

晨光是 ChinaUnix 专家博主，在 Unix/Linux 领域工作多年，在 ChinaUnix 发表了很多高质量的技术文章和 Linux 教学视频，深受广大网友喜爱。《开源安全运维平台——OSSIM 最佳实践》一书是他多年研究成果的总结，也是业界第一本关于开源安全运维的著作，书中采用了大量实例生动地讲解了 OSSIM 的安装和使用过程，深入浅出地分析了 OSSIM 关联分析等核心技术，引入了作者多年对 OSSIM 技术的研究成果和实践经验，这对于开阔读者眼界，提高技术水平将大有裨益。如果你从事系统运维，对网络安全感兴趣，我们强烈推荐此书。

ChinaUnix 社区

Linux 中国 推荐

从 2011 年起，我就在电信系统从事网络安全方面的工作，期间接触了不少企业客户，有央企、也有中小型公司。发现很多时候，企业在应对信息化普及所带来的变化时有些力不从心。较大的企业，其企业信息化也比较完善，除了大量的服务器、终端计算机、网络设备之

外，也有各种网络安全方面的专用设备和软件，但是，随着规模的扩大设备的管理、信息的归集会越来越步入低效，往往导致各种设备资产并不能及时有效地发挥作用。

面对企业的大量的 IT 设备管理包括多个方面，从资产管理、网络监控、漏洞管理、入侵检测等都有各种管理规范和相应的软硬件设备，那么增加这些管理系统是否又进一步加剧了信息臃肿呢？过去的做法是，针对每个细分都有一套乃至几套系统来管理，而这些系统之间并不能互相协调、信息共享，往往导致自相矛盾，让管理人员无所适从。当时，似乎并没有一个可以全面地、可靠地解决这些问题的方案。

举个例子，某国有大型银行，其为了应对网络安全风险，除了防火墙之外，还专门部署了 IDS 和 IPS 设备，但是随后发现各种事件、消息如洪水般涌来，将真正有价值的信息都淹没在了各种信息噪音之中。由于并不能针对企业实际的情况有效地降低无关或常规信息的干扰，导致每天发送的例行报告，也就真的成为了“例行”，从而只是增加了收件箱中的某个文件夹的未读邮件的数字而已。

那么，如何从纷杂的信息中及时准确地将重点的信息撷取出来？如何将各个设备、功能从各个方面，一致而完整地联系起来？

有幸认识了晨光老师，听他深入浅出地介绍了 OSSIM 系统，才发现这样的一套开源解决方案，恰恰满足了大部分企业在这方面的需求。OSSIM 是开源软件，在没有接触 OSSIM 之前，很多人会对它抱有一些疑虑，担心它的健壮性不足，担心它的功能不够全面，甚至担心它一如很多开源软件那样丑陋。但是 OSSIM 让我一个在开源圈混迹了多年的老兵也很吃惊，其表现绝对可以令人眼前一亮。那么具体 OSSIM 是怎么样的呢？这个问题可不是一两句话能说明白的，想详尽了解 OSSIM 的读者，请阅读这本晨光老师的力作吧！

王兴宇 Linux 中国（https://linux.cn/）创始人、前中国电信高级专家

开源中国推荐

接到晨光让我为他的新作写序的邀请后，我诚惶诚恐，尽管已经在技术圈里瞎混了十几载。但是运维对我来说既熟悉又陌生，虽然做开发，但几乎天天都要接触运维的工作，从服务器安装、应用环境安装到应用部署，以及后期的维护、扩容和安全等等方面，特别一开始做开源中国网站时，更是事无巨细、亲力亲为；但是跟晨光接触两年来，深感自己所做的这些是多么的微不足道。

从自己的从业经验来看，运维着实是一件非常专业的工作，而且要求经验必须非常丰富，才能从各种五花八门的现象中进行问题定位，从海量日志中分析问题，从而制定行之有效的解决问题的方案。当一个系统规模不断扩大的过程中，运维工作日趋重要。

在开源领域中有大量跟运维相关的开源软件，光开源中国网站就收录了运维相关的工具

超过 400 款，而 OSSIM 就是其中非常优秀的一款。OSSIM 是目前一个非常流行和完整的开源安全架构体系。OSSIM 通过将开源产品进行集成，从而提供一种能够实现安全监控功能的基础平台。它的目的是提供一种集中式、有组织的、能够更好地进行监测和显示的框架式系统。晨光的这本书从 OSSIM 的架构原理、安装部署，再到内部架构、高性能部署以及应用场景的实战等方面进行非常详细的讲解，不仅对软件的初学者适用，而且对经验丰富的工程师都有非常高的参考价值。

从此书的篇幅便知运维工作之复杂，唯有孜孜以求方能在强手之林有立锥之地，与君共勉。

红薯 开源中国办公室

前　言

为什么要写作本书

1. 现状

日常工作中，运维人员大部分时间和精力都用于处理简单、重复的问题，由于故障预警机制不完善，往往故障发生后才会进行处理，运维人员经常处于被动“救火”状态。

没有高效的管理工具支持，就很难快速处理故障。市面上有很多运维监控工具，例如商业版的 Solarwinds、ManageEngine 以及 WhatsUp 等，开源的 MRTG、Nagios、Cacti、Zabbix、OpenNMS、Ganglia 等。由于它们彼此之间没有联系，即便部署了这些工具，很多运维人员并没有从中真正解脱出来，成千上万条警告信息堆积在一起，很难识别问题的根源，结果被海量日志所淹没，无法解脱出来。

另外，在传统运维环境中，当查看各种监控系统时需要多次登录，查看繁多的界面，更新管理绝大多数工作主要是手工操作，即使一个简单的系统变更，也需要运维人员逐一登录系统，若遇到问题，管理员便会在各种平台间来回查询，或靠人肉方式搜索故障关键词，不断地重复着这种工作方式。企业需要一种集成安全的运维平台，满足专业化、标准化和流程化的需要来实现运维工作的自动化管理，通过关联分析及时发现故障隐患。

2. 手工整合的演化过程

在人工管理初期，主要依靠一些简单的 Shell 脚本完成一些基础工作，后来虽然采用 Cacti 来做性能监控，Nagios 做主机监控、PHP+SSH 等方式进行管理，但各种运维工具仍无法实现数据共享，此时整个防御体系面对网络威胁“反应迟钝”，每当故障来袭，总是“马后炮”，难以查找攻击者的踪迹，就好像一个人总被蚊子叮咬，想打蚊子可手眼又跟不上的感觉。

经过分析后，开始尝试将资产管理模块、入侵检测模块、流量监控模块、漏洞扫描模块集成到一台服务器中进行统一管理，实现了标准化日志、统一处理等任务，在系统改造中以下问题尤为突出：

- 安装时软件依赖问题难以解决。
- 各子系统界面重复验证和界面风格不统一。
- 各子系统之间数据无法共享。
- 无法实现数据之间关联分析。
- 无法生成统一格式的报表。

- 缺乏统一的仪表板以展示重要信息。
- 系统维护难度增大。

将这些开源工具集成比较困难，该方案架构并不合理，出现了性能瓶颈，对于安全事件的关联分析、合规管理及知识库查询依然无法实现。

3. 终极工具——OSSIM 集成安全运维的平台

发现一个好的管理平台并不是偶然，管理员从最原始的命令行的运维时代，进化到统一管理平台，的确要走很多弯路，其实这一过程就是普通管理员到专家的蜕变。只有经历过磨难的管理员才能深刻体会到这一点。一款优秀的安全运维平台，需要将事件与 IT 流程相关联，筛选出运维人员最关心的事件，提高工作效率。

目前能满足上述要求的开源产品只有 OSSIM 系统，它是由 AlienVault 公司开发的，现分为开源 OSSIM 和商业版 USM 两种，通过该平台实现对用户操作规范的约束和对计算机资源进行监控，包括服务器、数据库、中间件、存储备份、网络基础设施，通过自动监控管理平台实现故障综合处理和集中管理，能够为您的网络构建起一套敏感的、全方位的中枢神经系统，达到感知网络威胁的效果。

创作过程

说起来，我和 OSSIM 还是挺有缘分。研究生时曾开发过开源统一安全管理平台项目，主要目标是将不同网络设备和服务器的日志，通过标准化转化为事件，然后统一进行日志分析与设备联动。在完成这个项目过程中，主要参考 OSSIM 源代码，先后尝试了基于统计、基于距离和基于决策树的算法，攻破了网络安全事件聚合的难题。

这些年先后为几十家单位成功部署了 OSSIM 系统，并提供技术支持。在 OSSIM 项目实施过程中不断总结遇到的各种问题，经过三年的技术沉淀与积累，目前已经撰写出 600 多页的 OSSIM 应用教程，但是这些零散的手稿不成体系。从 2015 年初，开始将这些系统部署的经验进行合理组织，全书规划成三篇，共 10 章内容，这些内容包含 OSSIM 系统的各种知识和技巧，使读者今后再遇到问题能够举一反三。即使 OSSIM 更新升级后，读者也能结合书中介绍的概念和操作方法，同样能够掌握，那么本书的目标就达到了。

从事 IT 工作的人都比较忙，很少有完整的时间能清闲下来，对于一般人而言没有时间就是最好的幌子，而善于利用时间的人往往能够利用各种间隙进行创作构思。在本书创作中并不是一帆风顺，有时候为了验证一个技术问题需要反复实验，为了一句话需要经过反复推敲。

初稿出炉，必须经过不断修改润色才适合阅读。本书刚刚写完时才 500 多页，但是在一遍又一遍的修改笔误和错别字之后，萌发出新的想法，每复查一遍，我都会对原稿做一些改动，数量上要数第一次改动扩充最大，以后逐渐减少，直到满意为止。

本书不是什么神功秘籍，无法让你在短时间内从一个小白变成一个牛人。书中以 OSSIM 4 平台为基础进行讲解，将各种开源软件合理地融入进来，并把本人多年 OSSIM 实施经验以案例的形式表达出来。学习 OSSIM 的道路并不是一帆风顺，希望读者朋友再遇到困难时，本书能够为您答疑解惑。

篇章结构

书的结构好比框架，而内容则是具体组成元素，本书采用了文字、图表和范例等形式，将 OSSIM 复杂的结构和工作流程直观地展现给读者。全书分为三部分，共 10 章。

1. 基础篇

第 1 章：本章从 OSSIM 起源讲起，介绍了目前运维人员现状，逐步谈到应用 SIEM 的必要性，进而介绍 OSSIM 架构与组成原理，另外还介绍了基于插件的日志采集思路，提出标准化安全事件的全新理念，详细分析了 OSSIM 的高可用架构与实现方法。

第 2 章：本章从 OSSIM 实施关键要素、安装策略、硬件选型开始，深入分析单机部署，分布式体系、传感器设置等重要安装工作。分析安装过程以图文并茂的方式，指出了系统配置过程，包括实体机、虚拟机不同环境中的安装方法及注意事项。最后重点分析了 SIEM 事件控制台的使用和事件过滤方法。

2. 提高篇

第 3 章：本章对于 OSSIM 开发人员很有帮助，除了介绍 OSSIM 数据库组成、表结构，以及系统迁移备份等技巧以外，还包括各种常见 MySQL 故障等内容。

第 4 章：本章从关联分析基础讲起，逐步深入到 OSSIM 安全事件提取过程，介绍了常用的关联分析算法。还对报警事件的聚合原理做了详细分析，并结合 OSSIM 现状采用多个实例讲解关联规则和自定义策略的使用方法。

第 5 章：本章主要介绍各种 OSSIM 系统中的监控调试工具的使用，以及系统瓶颈的诊断方法。

第 6 章：本章重点介绍 Snort 原理和预处理程序发挥的作用，包括 Snort 报警方法。深入分析 Snort 规则编写在 OSSIM 中的应用技巧以及网络异常行为分析方法。

3. 实战篇

第 7 章：本章从日志标准化和收集分析方法讲起，详细分析各种服务、网络设备所产生的日志，包括 Apache、FTP、Squid、DHCP 等，并通过实例详细介绍 OSSIM 插件开发过程。

第 8 章：本章讲解 NetFlow 进行异常流量分析的方法，包括 NetFlow 数据采集和过滤方法，介绍了分布式环境中，利用 NetFlow 监测异常流量的技巧，同时针对 OSSIM 中 Ntop、

Nagios、NetFlow 三种检测工具的使用方法进行了对比。最后还介绍了 Cacti 和 Zabbix 第三方开源监控软件集成的方法。

第 9 章：本章从 OSSIM 控制管理中心角色权限控制讲起，全面介绍了 OSSIM Web UI 的结构，讲解了 OSSEC 日志分析工具的配置使用和 A gent 的安装方法。介绍了 OSSIM 中管理网络资产的实例，并对 OpenVAS 扫描模块、脚本以及规则做了深入分析。展示了多个利用 OSSIM 进行高级攻击检测的实例，以及利用 OSSIM 进行合规管理和系统统一报表输出的方法。

第 10 章：本章主要讲解基于 Web 方式下的抓包及数据包过滤方法，并采用该工具远程解决网络故障的方法，重点介绍了 tshark、tcpdump 等抓包工具的高级使用方法，最后以一个典型 IE 浏览器的 0 day 漏洞攻击的实例来检验这种工具所发挥的作用。

本书约定

（1）关于版本

本书软件的安装环境为 Debian Linux 6.0（Squeeze），内核为 2.6.32。在安装其他软件时，必须符合该版本要求。

（2）关于菜单的描述

OSSIM 的前台界面复杂，书中经常会用一串带箭头的单词表达菜单的路径，例如 Web UI 的 Dashboards→Overview→Executive，表示 Web 界面下鼠标依次经过菜单 Dashboards、Overview，最后到达 Executive 仪表板。

（3）路径问题

本书中除特别说明，所涉及路径均指在 OSSIM 系统下的路径，而不是其他的 Linux 发行版。终端控制台指通过 root 登录系统，然后输入“ossim-setup”启动 OSSIM 终端控制台的界面，如图 1 所示。

图 1　终端控制台

在终端控制台下，选择 Jailbreak System 菜单就能进入 Root shell，登录日志会保存在

/var/log/ossim/root_access.log 文件中。

（4）SIEM 事件分析控制台

书中的 SIEM 控制台是指通过 Web UI 进入系统，在菜单 Analysis→SIEM 下的界面，如图 2 所示。

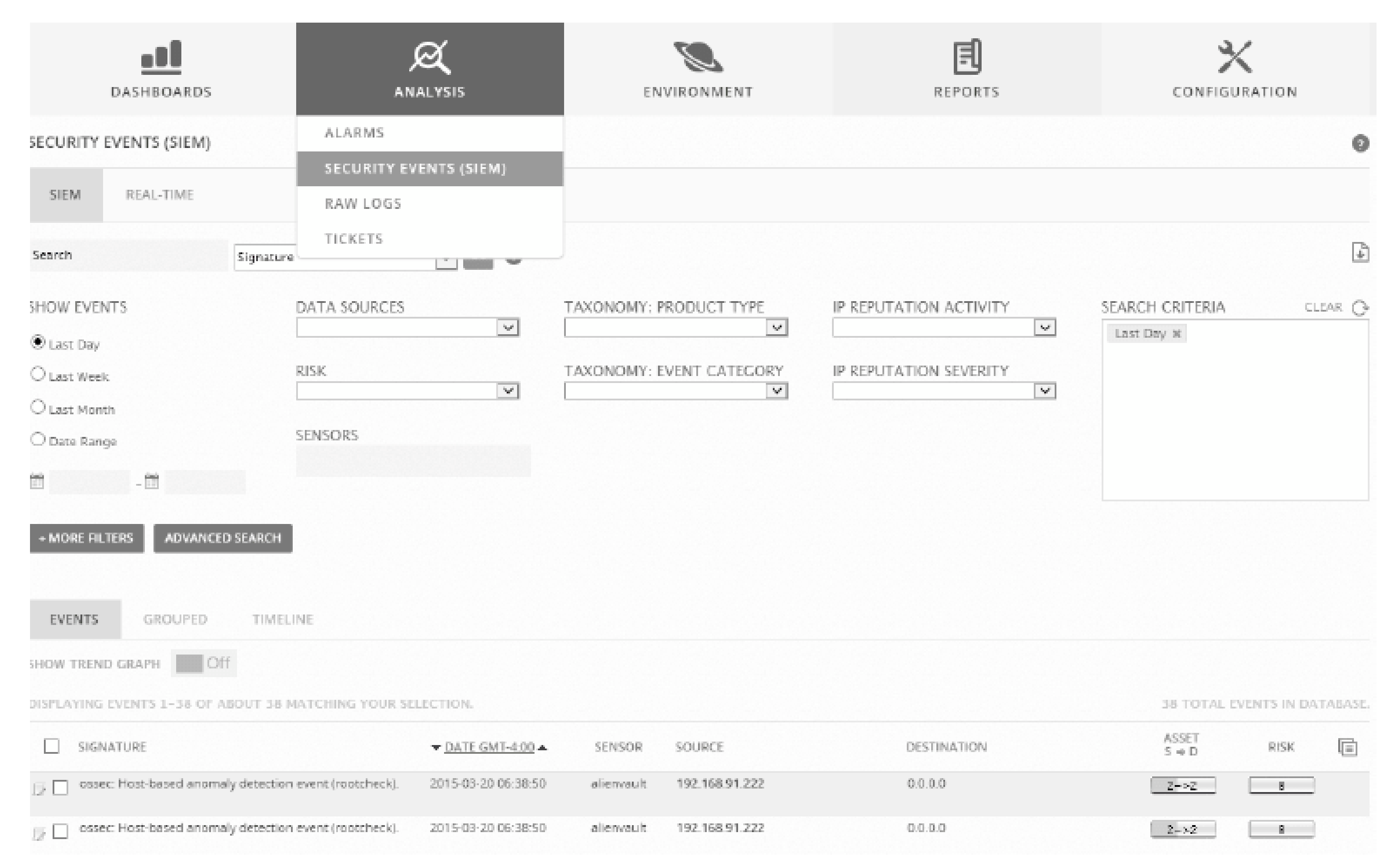


图 2　SIEM 事件分析控制台

（5）关于 OSSIM Server 端与 Sensor 端的约定

本书各章中讲述的 OSSIM Server 端，是指通过 Alienvault USM 安装的系统，包括 OSSIM 四大组件，Sensor 端是通过 Alienvault Sensor 安装的系统。

（6）关于地图显示问题

所有地图信息引自谷歌地图，大家在做实验前确保能连上谷歌地图，而且使用系统中 OTX，前提条件也需要能连接到谷歌。

（7）浏览器约定

OSSIM Web UI 适合采用 Safari 7.0 以上、Google Chrome 44.0 以上、IE 10.0 以上浏览器访问。

本书读者对象

本书主要面向以下类型读者：

- 互联网和安全行业的系统安全从业人员。
- 银行、证券和保险行业 IT 运维人员。
- 政府、高校和科研机构等单位 IT 运维人员。

光盘内容

本书配套光盘包括：OSSIM 入门多媒体教程、OSSIM 安装 ISO、OSSIM 源码三部分内容，其中视频内容有以下章节：

- 第一集：OSSIM 的由来及应用部署
- 第二集：网络威胁感知技术探讨
- 第三集：OSSIM 单机部署安装与分布式安装
- 第四集：OSSIM 仪表盘操作初步
- 第五集：SIEM 控制台与 Alarm 事件告警解析
- 第六集：资产管理与漏洞扫描
- 第七集：OpenVAS 组成及升级实践
- 第八集：NetFlow 应用
- 第九集：OSSIM 权限设置与策略管理
- 第十集：用 OSSIM 发现蠕虫攻击
- 第十一集：报表合规管理
- 第十二集：命令行模式下控制台综合管理

关于作者

李晨光，毕业于中国科学院研究生院，目前就职于世界 500 强企业，资深网络架构师、51CTO 学院讲师、IBM 精英讲师、UNIX/Linux 系统安全专家，现任中国计算机学会（CCF）高级会员；在国内《计算机安全》、《程序员》、《计算机世界》、《网络运维与管理》、《黑客防线》等专业杂志发表论文六十余篇。曾独著畅销书《Linux 企业应用案例精解》、《Linux 企业应用案例精解第 2 版》，《Unix/Linux 网络日志分析与流量监控》等经典学习教程，均被中科院图书馆、国内重点高校图书馆和国立台湾大学图书馆等 200 多家图书馆收藏。《Unix/Linux 网络日志分析与流量监控》一书，于 2015 年获最受读者喜爱的本版类图书奖。

本人经常受邀在国内系统架构师大会和网络信息安全大会发表技术演讲，2012 年担任中国系统架构师大会（SACC）运维开发专场嘉宾主持人。2013 年在 IT168 举办企业内网信息安全实践沙龙活动中发表技术演讲。2014（第十届）中国网络主管论坛北京站发表技术演讲。2014 年《网络运维与管理》杂志对本人进行独家专访并刊发于 13 期杂志中、2015 年 4 月在 WOT 互联网运维与开发者大会发表技术演讲，如图 3 所示。

图 3　作者在各种全国大会中发表技术演讲

支持与勘误

由于 OSSIM 本身结构复杂，知识点众多，在本书撰写过程中难免有所疏漏，希望广大读者能把问题反馈给笔者，本人不胜感激。为了方便读者学习实践，书中涉及所有软件和实验环境都已发布在作者博客 http://chenguang.blog.51cto.com/350944/1679097，在此博客中的 OSSIM 专栏包含了大量实战经验，大家可以一边阅读本书，一般参考博客，互为印证，如有问题大家可以留言，我将定期为读者解答。

也欢迎读者加作者的微博：http://weibo.com/cgweb。

致谢

首先感谢我的父母多年来养育之恩，感谢我在各个求学阶段的老师们，感谢每一位读者，你们将是本书继续完善的新动力，尤其要感谢我的妻子，有了她精心的照顾，我才能全身心投入到创作中。最后要感谢清华大学出版社的编辑们，为了提升本书质量他们花费了大量心血。本书若有不足之处，敬请读者不吝指正。

李晨光

2016 年 1 月

目　录

第一篇　基础篇

第二篇 提高篇

第三篇　实战篇

第一篇

基 础 篇

Open Source Security
Information Management

Big Data

Cloud Computing

Internet of Things

第 1 章

◂ OSSIM架构与原理 ▸

从本章节可以学习到：

- SIEM 概念
- Snort+ACID 架构
- OSSIM 架构与原理
- OSSIM 组件（插件、代理、传感器、Web 前端、关联引擎、数据库）
- Agent 事件类型
- 插件归一化处理流程
- OSSIM 工作流程分析
- RabbitMQ 在 OSSIM 中的作用
- OSSIM 系统的 CA 中心
- OSSIM 计划任务

1.1 OSSIM 概况

OSSIM 即开源安全信息管理系统（Open Source Security Information Management），是一个目前流行的开源安全架构体系。OSSIM 通过将开源产品进行集成，从而提供一种能够实现安全监控功能的基础平台，目的是提供一种集中式，能够更好地进行监测和显示的框架式系统。

现在，网络威胁从传统的病毒进化到蠕虫、拒绝服务等恶意攻击，当今的网络威胁攻击复杂程度越来越高，已不再局限于传统病毒、木马，还包括僵尸网络、间谍软件、流氓软件、网络诈骗、垃圾邮件、蠕虫、网络钓鱼等，它们都严重威胁着网络安全。OSSIM 系统通过将开源工具进行融合，从而提供一种能够实现安全监控功能的一体化平台。

OSSIM 定位为一个集成解决方案，其目标并不是要开发一个新的系统，而是利用丰富的、强大的各种程序，包括 Suricata、Ntop、Spade（异常检测引擎）、Tcptrack（TCP 会话实时监控）、P0f、Arpwatch（MAC 异常检测）、OpenVAS（漏洞扫描）、Nagios（主机及服务可用性监控）、Nikto、RabbitMQ、Redis、Ansible、Ossec 及 RRD Tools（网络链路流量监控软件）、RRD Tool 等开源软件。

在保留原有功能的开放式环境下，将它们有机集成起来。OSSIM 项目的核心工作在于负责集成和关联各种产品提供的信息，同时进行相关功能的整合，如图 1-1 所示。OSSIM 能为

您的企业网打造一个从运维监控→事前预警→事后报警→SIEM 日志分析故障的这样一个快速解决问题的网络系统。

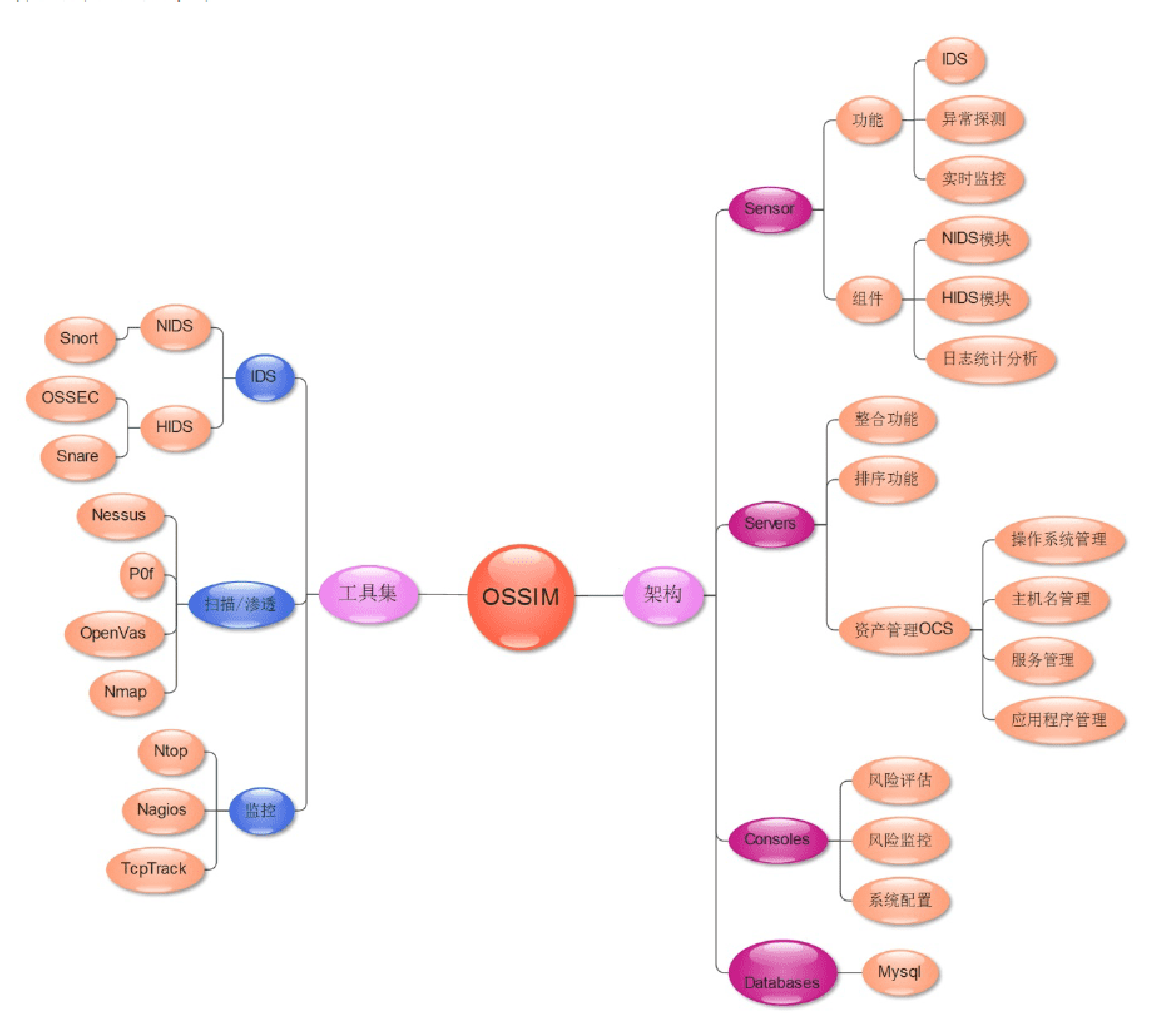


图 1-1　OSSIM 基本组成

1.1.1　从 SIM 到 OSSIM

IT 安全领域常面临着很多突出情况，必然会导致错综复杂的企业风险出现，这些风险既包括来自于企业外部的各种数不胜数的入侵行为，又包括来自于企业内部的各种违规操作以及信息的泄露。为了应对各种不断涌现的安全挑战，企业开始部署防火墙、IDS/IPS、漏洞扫描系统及各种防病毒系统等安全防御系统，这些设备确实起到了一定的安全防御作用，但同时却也引发出了新的问题——多套防御系统给企业网络环境带来了难以承受的复杂性。

目前，市场上没有哪家的系统能把上述这些设备都整合成统一分析系统。实际上即使部署了安防设备也只是安全孤岛，难以联动。此时 SIM 概念应运而生，它可以结合网络中心的现状，实现网络、系统、环境、安全等集中式的综合管理，最终实现网络中心的安全和维护保障安全防护水平的整体提升。从 2012 年的全球 IT 市场而言，对 SIM（Security Information Management）的需求十分强劲，例如 AlieVault 公司先后获得了三千多万的投资，目前旗下的 Alien Vault USM 产品开发迅速，每季度更新一个新版本。

OSSIM（Open Source+SIM），通过建立统一的综合管理平台体系，从而实现统一管理，将现有的监控和维护手段有机地联系起来，实现各个安全防御系统的统一协作，通过对已有安全信息的合理分析，可以预计网络威胁事件发生的情况，主要包括各种网络攻击的发生频率和规模，网络中攻击事件的严重程度。攻击事件越多，造成损害的风险越大，受感染主机就越多，造成主机数据泄漏，甚至网络中断的可能性越大。另外，僵尸网络越活跃（数量多，规模大），发生 DDOS 攻击，垃圾邮件泛滥的可能性越大。OSSIM 包含了 SOC（Security Operation Center）的主要内容，在分布式 OSSIM 系统中，可以快速地收集、分析和关联整个企业网中的安全设备日志信息。

1.1.2　安全信息和事件管理（SIEM）

日志管理是网络安全的基础工作之一，通常情况下日志收集并不涉及数据分析与挖掘，它只是从不同类型的系统、设备上收集并保存好日志，这里的系统和设备涵盖了交换机、防火墙、路由器、服务器（基于 UNIX/Linux/Windows 平台），以及应用程序（数据库、客户关系管理系统等），如图 1-2 所示。但日志收集不等于日志分析，收集工作仅为后期日志关联分析打下基础，日志分析实际上是日志消息的分析和处理，实践工作中需要在大量看似正常的日志中快速筛选出可疑的网络事件，以便进一步调查分析，例如寻找拒绝服务或者蠕虫病毒之类特定威胁等。

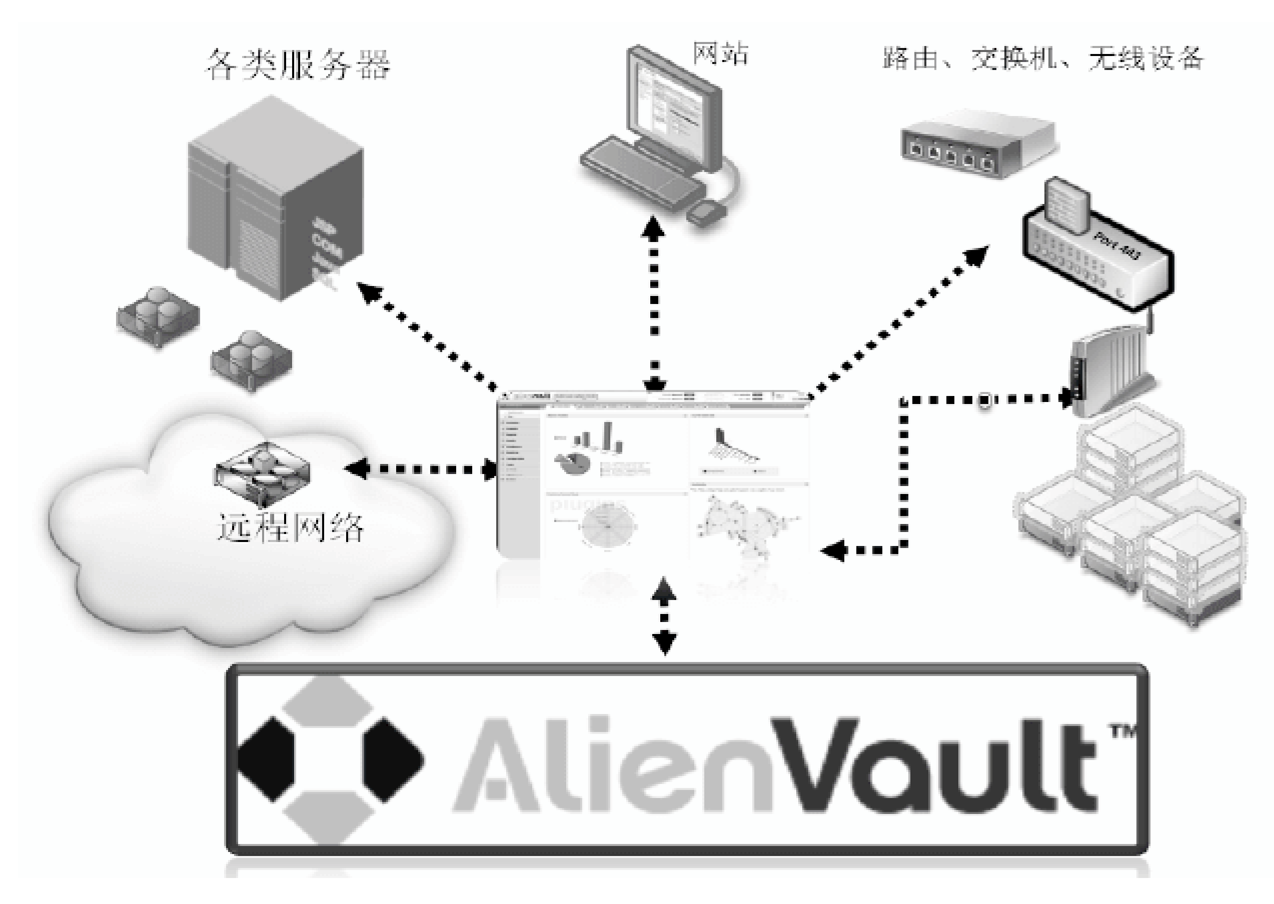


图 1-2　OSSIM 日志收集途径

过去被称为安全事件管理（SEM）或者称之为安全信息管理（SIM），现在称为 SIEM（Security Information and Event Management），它分为商业的解决方案和开源的两种，本章介绍的 OSSIM 系统就是安全信息与事件管理（SIEM）的开源解决方案。SIEM 通过相关事件、用户、系统、数据、风险和准确状态对策信息，从而让用户及时将各个安全节点联系起来，及时定位攻击，准确了解全网的安全态势。安全态势的数据主要来自网络中的安全设备（如防火墙、IOS/IPS、蜜网等）、网络设备（如路由器、交换机）、服务和应用。主要采集数据包括：网络流量（大小、流量成分）、关键网络设备（CPU 利用率、端口利用率）、入侵事件的数量

和严重等级、网络性能等数据。

OSSIM 是开源的 SIM，其核心仍是依靠 SIEM，主要优点是通过有关事件、数据、风险等信息，实时了解全网威胁态势。OSSIM 为了适应在大型网络中整合与分析，从各种应用程序和设备收集日志，它一开始设计成分布式结构，通过对网络安全态势的充分了解，将各个监控节点联系起来，形成一个数据链，经过关联分析就能对黑客攻击进行分析，进而能发出预警。目前国内很多企业信息化部署的安全防线，在很大程度上仍属于“事已发而后知”的状况。

针对病毒、木马、网络攻击，基本都采用“兵来将挡、水来土掩”的方式。虽然在实施拦截、隔离、清除的具体方式方法上各有千秋，但受限于技术条件，这些安全领域的企业无法做到提前预警，更谈不上防患于未然。

SIEM 的亮点是应对大数据时代的挑战。对于这一亮点应该比提前预警更容易理解，也更能够为客户企业所接受。无论是个人用户还是企业客户，在使用安全产品时几乎都遇到过相同的烦恼——对病毒库的频繁升级。在云时代未到来之前，这几乎是一道迈不过去的坎。如今，“云查杀”正在取代传统的单机病毒库，将数据及时反馈到云端服务器，利用更加全面的病毒库资源，更准确地提供解决方案并进一步扩充病毒数据库。OSSIM 中提供的 SIEM 这一特点无疑体现出了云时代的一大特征，不仅能够每天处理企业中庞大的各类网络事件，而且能够将这些事件与威胁、对策和用户身份信息相关联，以提供准确的、切实可行的智能信息。

1.1.3 OSSIM 的前世今生

Linux/Unix 中常用 Snort 来构建 IDS 系统，最常见的是 Snort+BASE 的架构，OSSIM 设计之初为 Snort+ACID 架构，如图 1-3 所示。

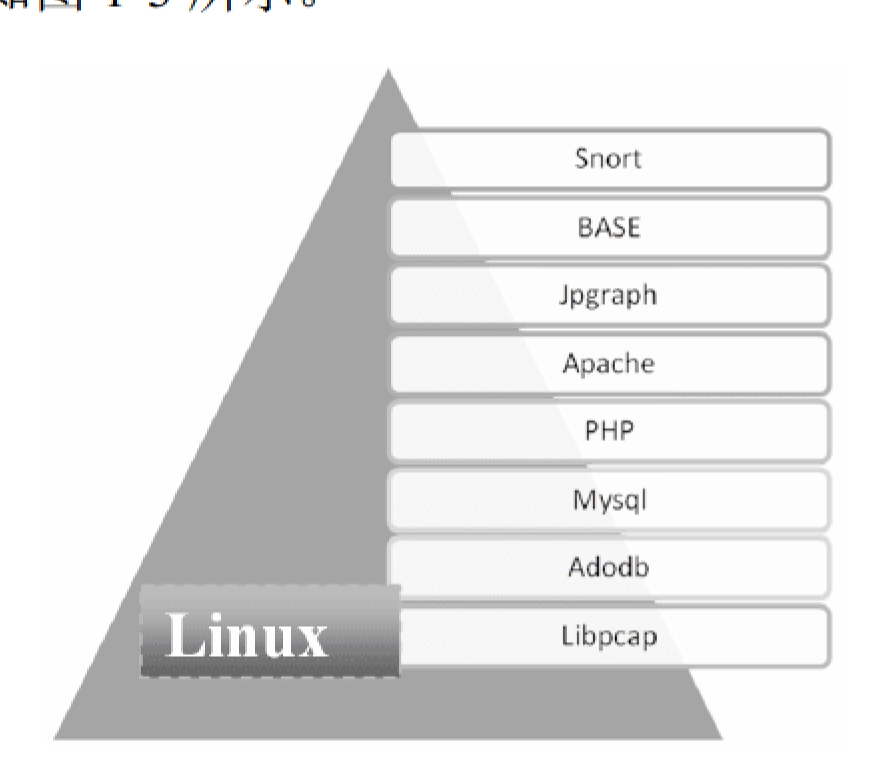


图 1-3 轻量级 IDS：Snort+ACID 架构

它们的组件包括：Linux、Apache、MySQL、PHP、Libpcap、Adodb Snort、Base、Jpgraph。其系统结构基于 Snort 和 Base 的 IDS 系统，通常采用“传感器+数据库+分析平台”三层架构，这三层可以装在单机，也可以分布式安装，这取决于实际网络环境。Snort+Libpcap 构成了最原始的 Sensor 传感器。如果你需要单独架设这一传感器，需要准备以下软件（版本以当前为准）：

- zlib-1.2.7.tar.gz

- libpcap-1.0.0.tar.gz
- libxml2-2.6.19.tar.gz
- libpng-1.2.44.tar.gz
- gd-2.0.33.tar.gz
- jpegsrc.v7.tar.gz
- mysql-5.0.22.tar.gz
- DBD-mysql-3.0008.tar.gz
- httpd-2.2.14.tar.gz
- php-5.2.9.tar.bz2
- pcre-8.00.tar.gz
- snort-2.8.3.1.tar.gz
- snortrules-snapshot-2.8.tar.gz
- snortrules-snapshot-CURRENT.tar.gz
- jpgraph-3.0.6.tar.bz2
- adodb498.tgz
- acid-0.9.6b23.tar.gz

正确安装 Snort 服务器组件包括 OpenSSL、Stunnel、OpenSSH、Apache、MySQL（Server、Client）、GD、PHP、ADODB、Libpcap。Snort 系统安装难点在于其众多组件的安装，由于数量多安装步骤繁杂，安装时需要按照先后次序操作（操作者必须了解每一个组件的作用），因为有些软件包必须要在另一个软件包安装完后才能进行编译和安装，因此满足软件包的依赖关系至关重要。Snort 系统安装流程如下（具体每种软件包安装的细节见作者博客）：

1. 安装 OpenSSL

OpenSSL 在互联网上广泛应用在在线支付、网银、电商网站、门户网站、电子邮件当中。近期互联网安全协议 OpenSSL v1.0.1 到 1.0.1f 的密码算法库中发现了一个非常严重的漏洞（CVE-2014-0160），该 Bug（缺陷）会暴露加密流量的密钥、用户的名字和密码以及访问的内容，该漏洞被称之为 HeartBleed（心脏流血），建议用户安装最稳定的 OpenSSL0.9.80。

安装 OpenSSL 首先是下载正确的源代码，可以在下面这个网站得到源码：

```
http://www.openssl.org/source/或 ftp://ftp.openssl.org/source/
```

源码包下载后经过解压、预编译、编译，如果不出现依赖关系问题那么你可以进入下一环节。在源代码安装方式中，OpenSSL 会被安装到/usr/local/ssl 目录下，OpenSSL 由下面几个重要文件组成：

- Libcrypto.a：该文件是二进制文件格式，它是通用的密码加密程序，包含了加密密码（libDES、IDEA）、消息摘要（MD5、SHA）、公共密钥（RSA、DSA）X.509 证书等。
- Libssl.a：包含了 SSL 和 TLS 的代码。

我们还需要让其他软件包知道 Openssl 软件包更新了。编辑/etc/ld.so.conf，在文件中添加一行：

```
/usr/local/lib
```

然后，保存退出，运行 ldconfig 命令更新新的共享库。

查看安装版本：

```
#openssl version -a
```

2. 安装 OpenSSH

我们利用 OpenSSH 来保证远程维护的安全，OpenSSH 通过证书和会话数据加密来维护系统安全，它包括以下内容：

- Ssh：可以实现两个主机之间的安全、加密通信。
- Scp：用于实现两台主机之间安全的文件传输，但 scp 依赖于 ssh 的验证和加密手段。
- Sshd：这是 ssh 的守护进程（服务进程），负责监听 ssh 客户端发来的链接请求。
- Sftp：用来实现文件安全传输，它采用加密的 ssh 传输，可以完成普通 FTP 的所有功能。
- Sftp-server：这是用来向 sshd 提供 sftp 协议信息。
- Ssh-agent：身份验证代理，这个工具是用来为 RSA 公共密钥证书保存私有密钥。
- Ssh-add：用于向身份验证代理 ssh-agent 中添加新的 DSA 密钥信息。
- Ssh-keygen：用于为 ssh 生成身份验证密钥。

下载地址：http://www.openssh.com/portable.html。

OpenSSH 依赖于 OpenSSL 正常工作，它必须依靠 OpenSSL 加密库工作。截至 2015 年 4 月，最新的版本为 OpenSSH-6.8，当下载文件时会发现类似 openssh-XX.tar.gz.asc 的文件，每个源代码版本的文件都对应有个 asc 文件（.sig 文件作用和它类似），该 asc 文件主要是检验包的完整性，下面看 openssh-6.6p1.tar.gz 源码包中 asc 的例子：

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.22 (OpenBSD)

iQGlAwUAUyEVxtPl9WttkgOwAQr39wx/SkCPbWVsEG2Do7IXKg1FudhKIt8QGQZC
4bNvnwFbFZ/RNEv+BZRKH7AmgNLXn6/VTlRkifF3osrvJ6gh8EfQMkNPB5WcRz5h
c4KdAcL6u/KFxku0x/hjqUaAV98OHlUFvAmGv3/YTL8n14uRBRlrVCnG5q1LmIv7
piBnb+FrAivrNV+PUMkkrC6XKrfR2ns+ZtfgwBQGQ/VuJTQZPOUMp/GLViHBxz4l
GOH52bolzAIunzBxXF9fWwFU8JFtCEharFr7jYuLQzViUPCaf3L9w3yPb2S24wnU
772CEyGbdCJVLmtD9ObaNKk0We6hdc6Dz+mx/JbHkoWSw0LYdHFbjFR75vjNENYh
evRgqLCP91micfhZDQqX6tOLnMfBjxCjHsCYvMlDACiEvxb1Ct6Fm9m1lXIGUsh3
ssScaCvQItMAvQdjQOdCXS7Zcucfzzx3Fd8Q2X/h2R81yYv6JjYQy7aBV/DaFDzP
/BhCwnq5sP6XW6rHRZGYXhMaAEjZml8ySRFEzW+esnGf9vSJgnQ+Kw==
=H2K/
-----END PGP SIGNATURE-----
```

下载的软件包都需通过 GnuPG 签名，一旦服务器被攻陷，这些源代码就有可能被修改，若文件的任何地方被修改，则签名文件就会改变，所以为了保证安全，花些时间检查你所下载的软件包是否完整，很有必要。

按照下面的步骤进行预编译、编译和安装：

```
#./configure; make; make install
```

接下来开始产生 DSA 密钥：

```
#ssh-keygen
```

一些 Linux 发行版可能会在连接到 zlib 压缩库的时候遇到错误，这时需要先安装 zlib 库（下载地址 http://www.zlib.net/）。

3. 安装 Libpcap

Snort 传感器上必须安装捕包工具，而 Snort 自身并没有捕包工具，它需要外部程序库 Libpcap 负责从网卡捕包，它不仅是为 Snort 程序服务，而且是由底层操作系统提供给其他应用程序的捕获原始包工具。下载地址 http://www.tcpdump.org/release/，最新版本 libpcap-1.7.3.tar.gz。Libpcap 缺点：

数据包的传输过程中函数调用和内存复制次数过多。

revfrom 是一个系统调用函数，每调用一次，它只向用户空间传递一个包，而系统调用实际上是一个中断，因此用传统的方法进行大流量捕包时，系统将会不断地进行中断处理和进程切换，那么严重时会导致 CPU 无暇顾及其他任务。

4. 安装 MySQL 服务器和客户端

应用程序需要 MySQL 客户端和 MySQL 服务器建立远程连接。比如 Barnyard 这样的程序也需要向 MySQL 发送警报数据，因此必须安装 MySQL 客户端。接下来我们开始安装 MySQL 服务器和客户端。主要分为 5 步：

- 下载。下载地址：http://dev.mysql.com/downloads/。最新版本为 5.6.24，大家可以下载源码包编译。
- 配置。
- 安全加固。
- 优化。
- 创建入侵数据库。这一步你需要创建一个用于存放事件信息的数据库，接着需要为 Snort 传感器创建一个用户来登录数据库。

5. 安装 NTP

网络时间协议 NTP，用于在多台物理分散的设备之间同步时间，而同步时间是事件关联分析必备工作。

6. 安装 Snort 和检测规则

对于规则安装此处省略，详情参见本书 Snort 规则讲解的内容。Snort 下载地址：https://www.snort.org/downloads/。

7. 配置 snort.conf

Snort.conf 文件是配置 Snort 的重点，必须指定要监控的 IP 地址范围、启用的预处理程序以及使用的输出插件和采用的规则。

8. 运行 Snort

Snort 通过一些命令选项来控制，需要用命令行给 Snort 传递以下内容：

- Snort.conf 配置文件路径。
- 日志文件夹路径。
- 网络接口（作抓包用）。
- Snort 用户和组的 UID（用户 ID）和 GID（组 ID）。

当 Snort 正常运行后需要查看报警，后面介绍 Web 入侵分析控制台。

9. 安装 Apache

不管哪种 Web 入侵控制台，在前端须安装 LAMP 环境。Apache 下载地址：http://httpd.apache.org/。

10. 配置 mod_ssl

mod_ssl 是 Apache 和 OpenSSL 之间的接口，Apache 可以通过各种模块扩展功能。Apache 有几百种模块，mod_ssl 只是其中之一。

11. 安装 GD

如果想在 Snort 的前端控制台上看到各种图形显示，那么就必须安装 GD 库，它是一种图形库，可以让 PHP 绘制出各种复杂的图形，GD 库可以创建 JPG、PNG 和 BMP 图像，可以分析大量的数据集合，然后将这些数据绘制成图表，将图表动态地展现给管理员。如果打算发挥 GD 库的功能就必须有以下 3 个库的支持。

- Zlib 压缩库：zlib-1.2.7.tar.gz。
- LibPNG 创建 PNG 文件库：libpng-1.2.44.tar.gz。
- Jpegsrc JPEG 压缩库：jpegsrc.v8.tar.gz。

只有在以上三个库安装完毕之后，才能开始安装 GD 库，其下载路径：http://pkgs.org/download/php-gd。

12. 安装 PHP

Snort 服务器管理图形用户接口 ACID 采用 PHP 语言编写，主要用途是将传感器收集的入侵数据制作成动态 Web 页，而且查询接口也是 PHP 编写，故在 Snort+ACID 架构中 PHP 环境须安装完整。

13. 安装 ADODB

由于 PHP 不能直接访问 MySQL 数据，中间必须通过一个接口库，这个接口库就是 ADODB，它在两者间起到桥梁作用。它的位置通常都在 PHP 目录下，以 OSSIM 系统为例，其路径为/usr/share/php/adodb/，配置文件为/usr/share/php/adodb/adodb.inc.php。

14. 安装 ACID

ACID（入侵数据库分析控制台）是一个基于 PHP 的分析引擎，它通过 Web 界面来查看 Snort 所产生的安全事件。ACID 能运行在多种操作系统中，为了使用 ACID，用户系统中必须安装 Snort、Apache、MySQL、PHP，它们之间的关系如下：

- 当入侵者进入监控网段内并进行各种攻击行为，Snort 会根据规则检测到入侵行为，然后根据其配置文件/etc/snort/snort.conf 的配置，将告警信息记录到 MySQL 数据库。
- 用户使用 Web 浏览器连接到 Snort 服务器。
- PHP 连接到数据库，提取告警信息。
- 用户在浏览器中查看告警信息。

在 OSSIM 系统中这些工具包已经集成到系统中，普通用户用 ISO 镜像即可快速安装，深入了解 OSSIM 之后可以手动安装 OSSIM，详细内容参考博文 http://chenguang.blog.51cto.com/350944/1691090。

为了监控完整，可根据网络分布情况，在多个网络关键节点上分别部署 IDS 传感器。当传感器 Snort 获得记录信息后有几种处理方式：

- 存储到本地日志。
- 转发到 Syslog。
- 存储到 MySQL。

Snort 日志记录仅包含网络数据包的原始信息，对于这些大量信息进行人工分析，显然不太可能，所以产生了一个能够操作查询数据库的分析平台，目前更新到 BASE（Basic Analysis and Security Engine，http://base.professionallyevil.com/，最新版本 1.4.5）版本。初学者独立架构这样一套复杂系统会遇到各种问题，有什么办法可以将这些组件一次性安装配置好，形成即装即用的产品呢？在 2003 年 8 月，OSSIM 的雏形便产生，经过十多年的演进，目前已发展成为一套功能齐全的安全管理与分析平台，其开发公司 Alienvault 在 2012 年 7 月获 3440 万美元融资，发展势头喜人。下面我们看看 OSSIM 各版本变迁，如表 1-1 所示。

表 1-1　OSSIM 版本变迁

年份	月份	OSSIM 版本
2003 年	8 月	OSSIM 0.1~0.4
	9 月	OSSIM 0.5
	10 月	OSSIM 0.6
	11 月	OSSIM 0.7
2004 年	1 月	OSSIM 0.8
2005 年	2 月	OSSIM 0.9.8
2006 年	6 月	OSSIM 0.9.9rc2
2007 年	6 月	OSSIM 0.9.9rc3
2008 年	2 月	OSSIM 0.9.9
2009 年	1 月	OSSIM 2.0
	7 月	OSSIM 2.1
2010 年	2 月	OSSIM 2.2
2011 年	7 月	OSSIM 2.3.1
	9 月	OSSIM 3.0
2012 年	1 月	OSSIM 3.1
	12 月	Alienvault OSSIM 4.1
	5 月	Alienvault　OSSIM 4.2
2013 年	11 月	Alienvault　OSSIM 4.3
	12 月	Alienvault　OSSIM 4.4
2014 年	3 月	Alienvault　OSSIM 4.5
	4 月	Alienvault　OSSIM 4.6
	5 月	Alienvault　OSSIM 4.7
	6 月	Alienvault　OSSIM 4.8
		Alienvault　OSSIM 4.9
	7 月	Alienvault　OSSIM 4.10
	9 月	Alienvault　OSSIM 4.11
	10 月	Alienvault　OSSIM 4.12
	11 月	Alienvault　OSSIM 4.13
	12 月	Alienvault　OSSIM 4.14
2015 年	1 月	Alienvault　OSSIM 4.15
	4 月	Alienvault　OSSIM 5.0
	6 月	Alienvault　OSSIM 5.0.3
	8 月	Alienvault　OSSIM 5.1.0

从上表可以清楚看出，OSSIM 系统开发从 2013 年底开始发力，以后开发速度越来越快，本书主要使用 2014 年的稳定版本 OSSIM 4.8 为平台进行讲解。

1.2 OSSIM 架构与组成

从架构上来看，OSSIM 系统是一个开放的框架，它的核心价值在于创新地集成各开源软件之所长，OSSIM Web UI 主要采用 B/S 架构，Web 服务器使用 Apache。OSSIM 系统结构示意图如图 1-4 所示。数据处理的层次感，在这张图中体现得淋漓尽致。

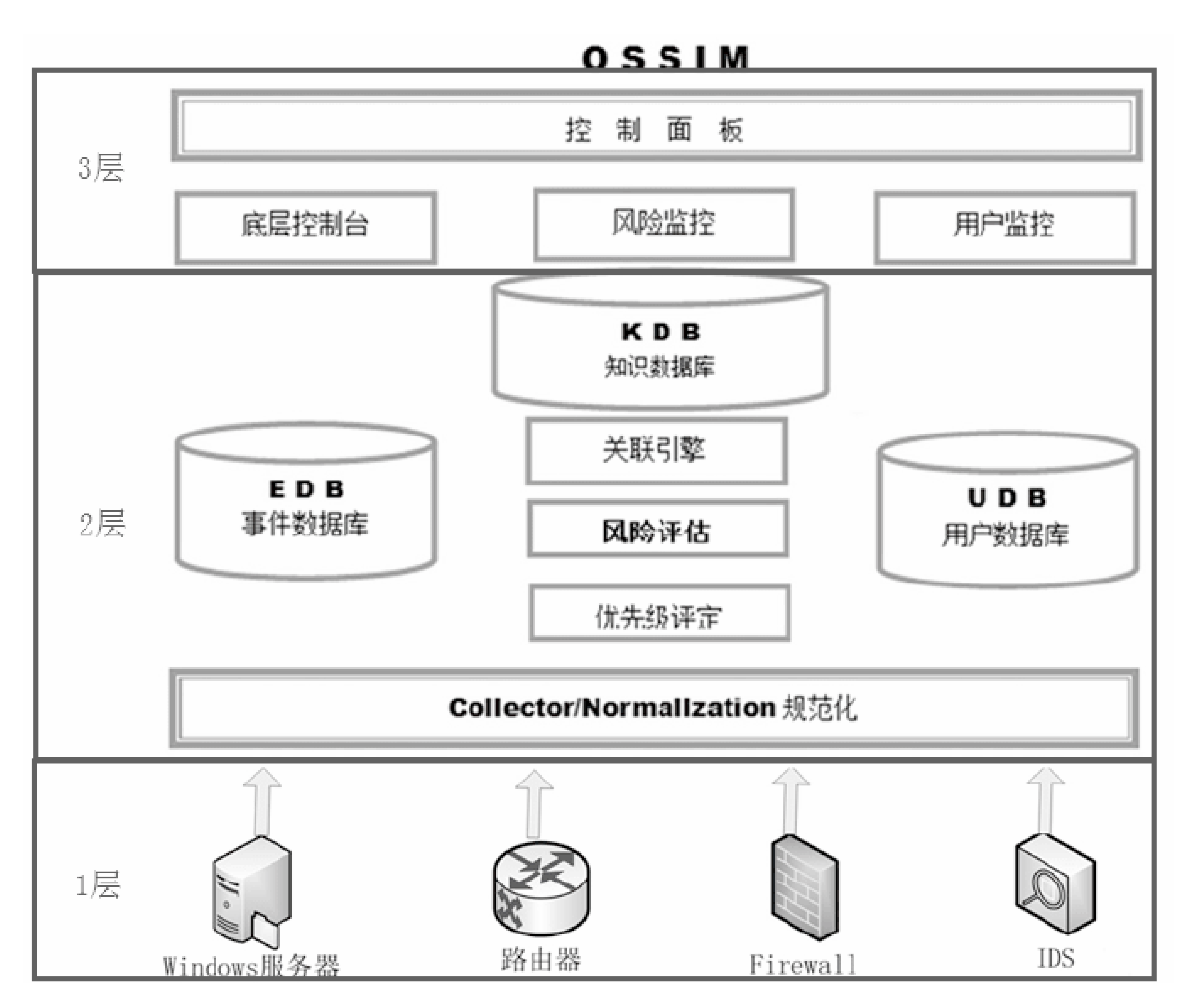


图 1-4 OSSIM 系统结构

如上图所示，OSSIM 系统结构包括三个层次：数据采集层、核心处理层和数据展现层。

第 1 层，属于数据采集层，使用各种采集技术采集流量信息、日志、各种资产信息，经过归一化处理后传入核心层。该层体现安全事件来源，入侵检测、防火墙、重要主机发出的日志都是安全事件来源，对于 OSSIM 而言，数据越多越好，如果没有收集足够的数据，OSSIM 就无法进行全面分析。它们按发出机制分为两类：模式侦查器和异常监控（两者都采集警告信息，功能互补），由它们采集的安全事件，再被 Agent 转换为统一的格式发到 OSSIM 服务器，这一层就是 Sensor 要完成的内容。

第 2 层，属于核心处理层，主要实现对各种数据的深入加工处理，包括运行监控、安全分析、策略管理、风险评估、关联分析、安全对象管理、脆弱性管理、事件管理、报表管理等。该层中 OSSIM Server 是主角，OSSIM 服务器主要功能是安全事件的集中，并对集中后的事件进行关联分析、风险评估及严重性标注等。所谓的集中就是以一种统一格式组织所有系统产生的安全事件告警信息（Alarms），并将所有的网络安全事件告警存储到数据库，这样就完成了对网络中所产生事件的一个庞大视图。系统通过事件序列关联和启发式算法关联来更好地识别误报和侦查攻击的能力。

OSSIM 本质上通过对各种探测器和监控产生的告警进行格式化处理，再进行关联分析，通过后期处理，能提高检测性能，即减少告警数量，减小关联引擎的压力，从整体上提高告警质量。

第 3 层，属于数据展现层，主要负责完成与用户之间的交互，达到安全预警和事件监控、安全运行监控、综合分析的统一展示，形式上以图形化方式展示给用户。Web 框架（Framework）控制台界面即 OSSIM 的 Web UI（Web User Interface，Web 用户界面），其实就是 OSSIM 系统对外的门户站点，它主要由仪表盘、SIEM 控制台、Alarm 控制台、资产漏洞扫描管理、可靠性监控、报表及系统策略等部分组成。

1.2.1　主要模块的关系

从使用的开发工具上看，OSSIM 系统主要采用了 PHP、Python、Perl、Erlang、Ruby、Ajax 和 C 这几种编程语言，从软件层面上看 OSSIM 框架系统包括五大模块：Agent 模块、Server 模块、Database 数据库模块、Frameworkd 模块以及 Framework 模块，逻辑结构如图 1-5 所示。

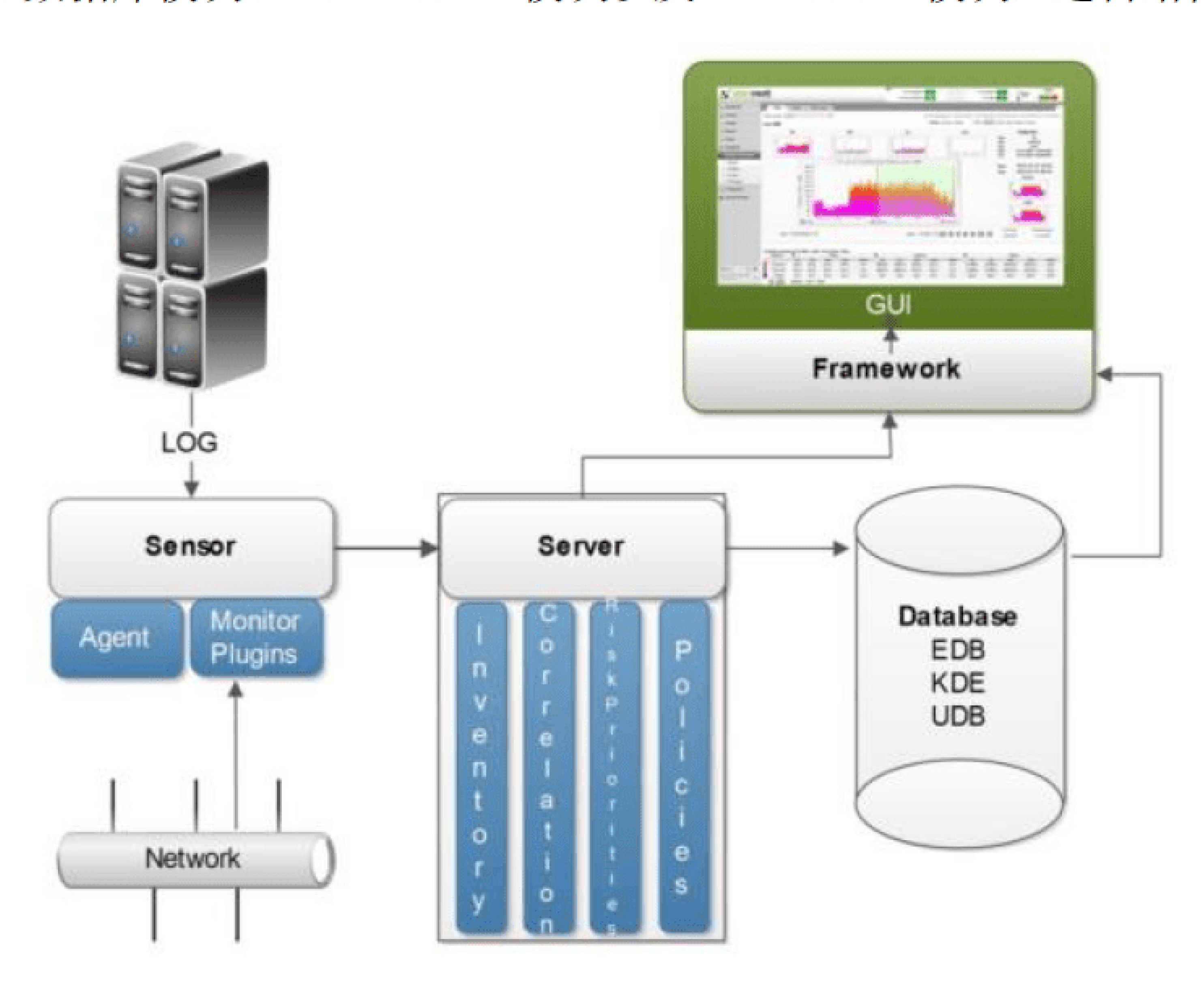


图 1-5　OSSIM 系统逻辑结构

OSSIM 五个模块之间的数据流向如图 1-6 所示：

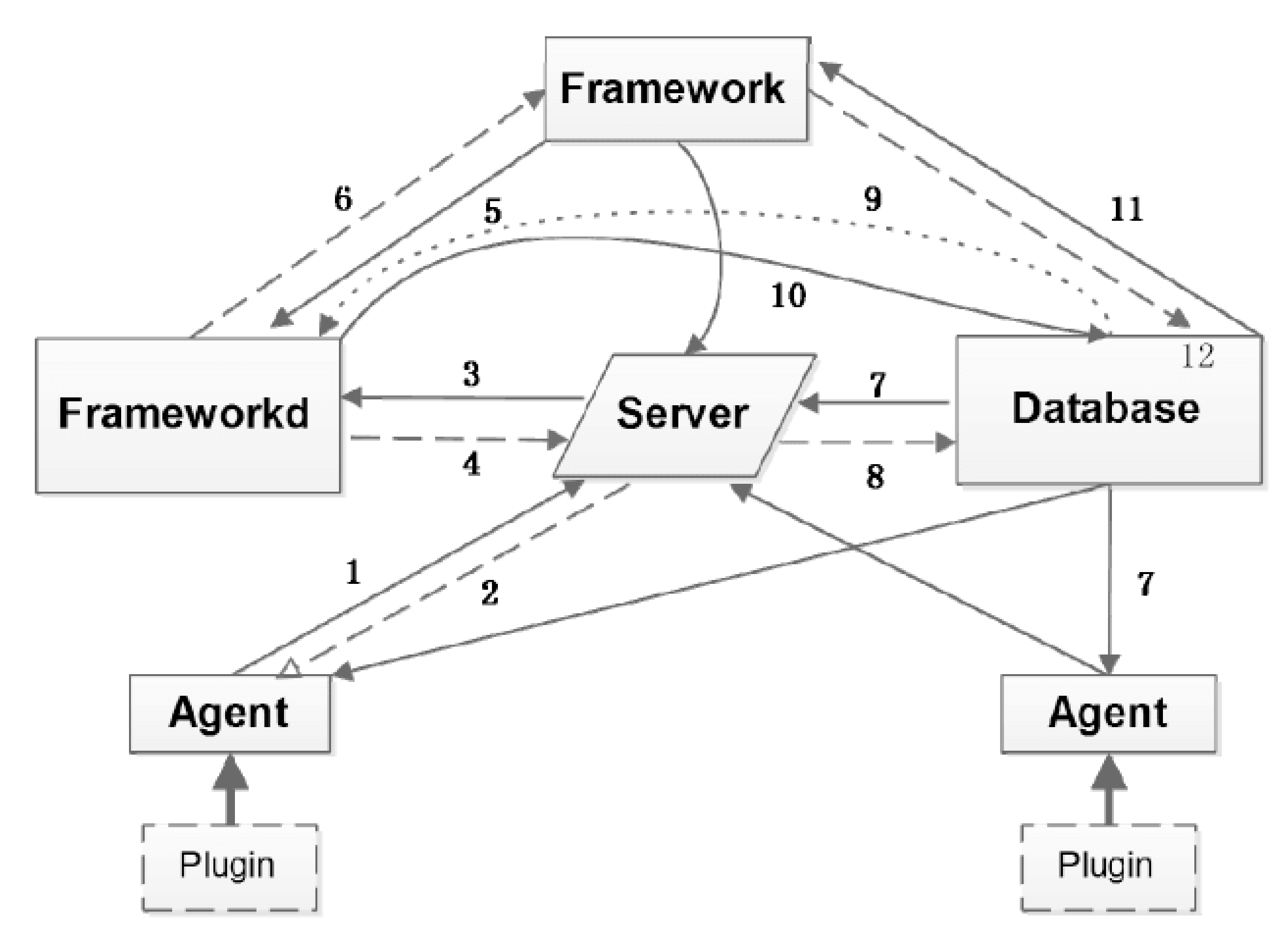


图 1-6　五大模块的数据流向

（1）Agent 至 Server：来自各个传感器的安全事件被对应 Agent 格式化后，以加密字符串传给 Server。

（2）Server 至 Agent：发送有关请求命令（request command），以字符串方式向 Agent 传送，主要是要求 Agent 完成插件的启动停止及获取信息等。

（3）Server 至 Frameworkd：发送请求命令，要求 Frameworkd 针对 Alarm 采取相应操作，例如执行外部程序或发出 Email 来通知管理员。

（4）Framework 至 Server：发送请求命令至 Server。要求 Server 通知 Agent 对插件（Plugins）进行启动、停止等操作。

（5）Framework 至 Frameworkd：发送请求命令，要求 Frameworkd 启动 OpenVAS 扫描进程。

（6）Frameworkd 至 Framework：传送 OpenVas 扫描结果在前端页面中显示。

（7）Database 至 Agent 和 Server：向 Agent 和 Server 提供数据。

（8）Server 至 Database：Server 需要将 Events、Alarms 等数据存入数据库并索引或更新操作。

（9）Database 至 Frameworkd：在 Frameworkd 中的 Openvas 扫描和动作需要用调用数据库里的数据。

（10）Frameworkd 至 Database：在 Frameworkd 执行过程中将 Openvas 扫描结果存入数据库。

（11）Database 至 Framework：PHP 页面显示需要调用数据库的告警事件。

（12）Framework 至 Database：用户参数设置信息需要存入数据库。

1.2.2　安全插件（Plugins）

OSSIM 4 系统中插件很多，可将它们分为采集（Collection）插件和监视（Monitor）插件。每个插件又细分为 ID 和 SID。采集插件主要通过 SNMP、Syslog、WMI 等协议进行采集，在

Sensor 中常见采集插件有 ossec-single-line、ssh、syslog、wmi-system-logger 等，其中 SNMP 与 WMI 协议需要 Agent 采集数据时主动进行所采集数据的抓取；Syslog 协议则被动接收采集数据。具体如图 1-7 所示。

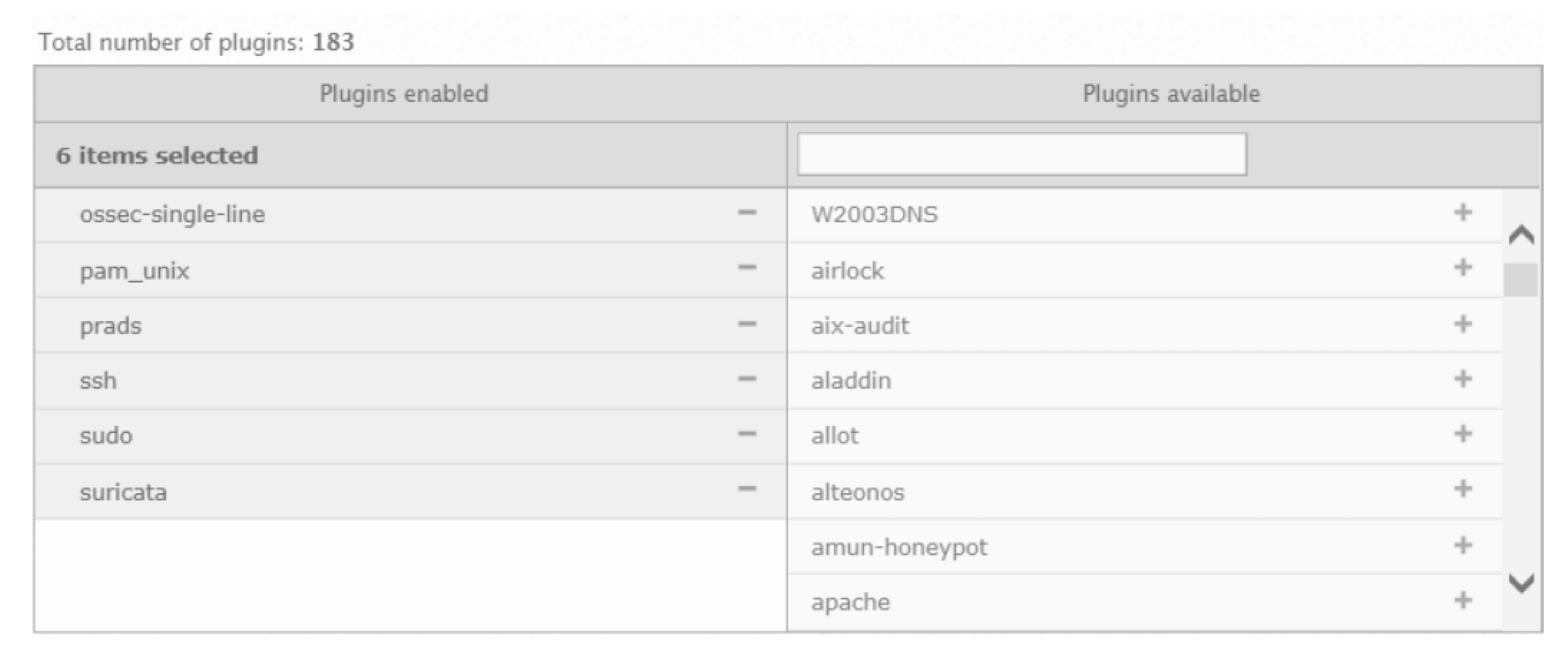


图 1-7　查看插件

监控插件包括 malwaredomainlist、nessus、nmap、ntop、ocs、ossim 等，如图 1-8 所示。

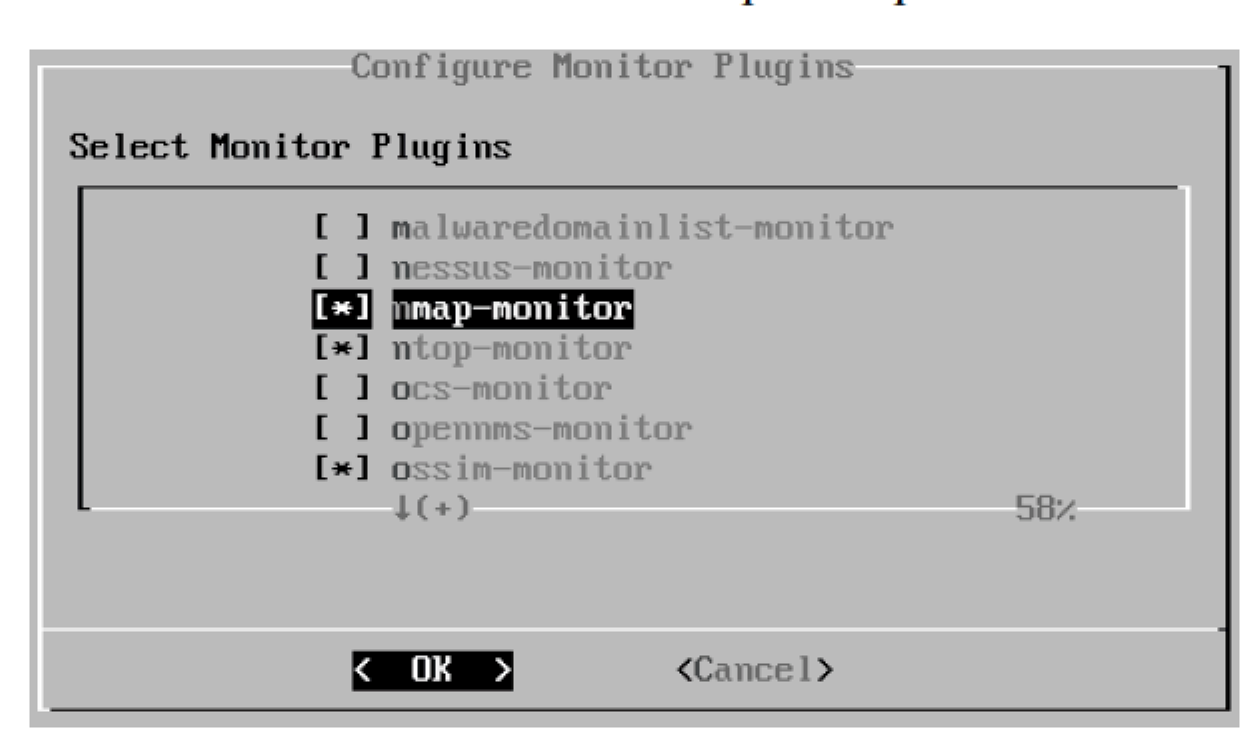


图 1-8　设置监控插件

当这些插件加载完毕之后，我们可以到 Web UI 中 Configuration→Deployment→Components→Sensors 栏目下的“Sensor Status”查看所添加的监控插件的工作状态，如图 1-9 所示。注意：在 OSSIM 5.x 版本中移除了 iphone、forensics-db-1、malwaredomainlist-monitor、motion、nessus-monitor ntop-monitor、snortunified 插件，OSSIM 4.X 中 p0f、PADS、ARPwatch 也被 PRADS 所取代，suricata 也被 Alienvault_NIDS 取代，ossec_sigle_line 被 Alienvault_HIDS 取代。

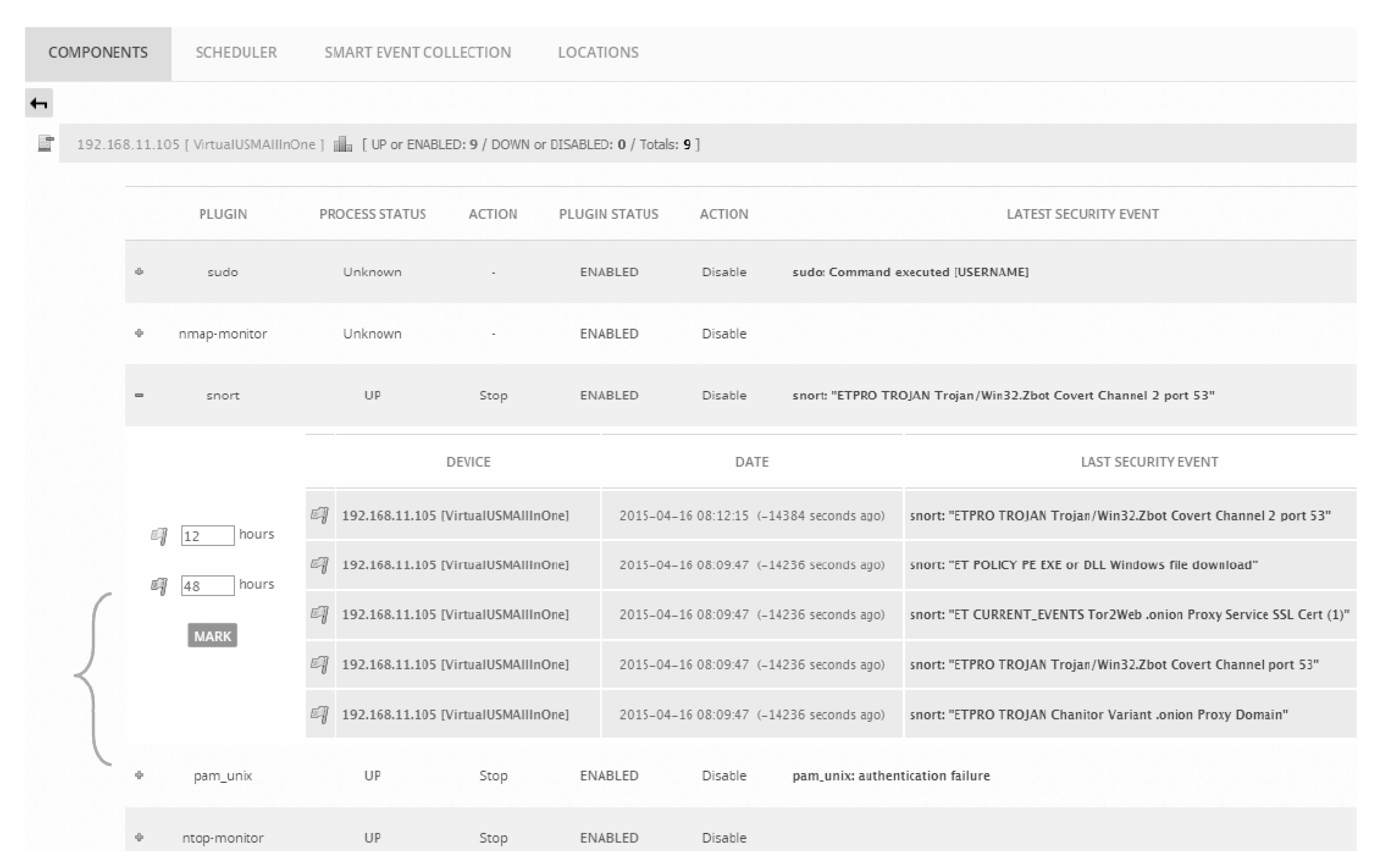


图 1-9　查看 Sensor 监控插件工作状态

UNIX/Linux 环境下，大部分系统都安装有 SNMP 与 Syslog。如果采集数据的目标系统为 Windows，那么考虑使用 WMI 协议，此时只需要在 Windows 上进行相关配置，以便能够远程访问，无须安装额外的工具软件。

1.2.3　采集与监控插件的区别

OSSIM 系统的 Sensor 端包含了采集（Collection）和监控（Monitor），这两类插件统称为安全插件，它们都安装在 Sensor 上。虽然它们都称为插件可工作原理却不同，检测插件（Detector）是检测器信息产生后，由代理自动向服务器发送，包括 Snort、Apache 等。而检测器插件需要主动采集安全设备接口上的信息，这类插件可分为 Snort、P0f、Prads、Arpwatch、Apache、Ssh、Sudo 等。

监控（Monitor）插件，必须由服务器主动发起查询请求。监控插件中定义了需要主动采集的安全设备接口，该模块接收控制中心发出的命令和查询，在 OSSIM 系统中典型 Monitor 插件有 Nmap、Nessus 等。读者可在 Alienvault 控制台的 Sensor 配置中（Configure Monitor Plugins）查看。OSSIM 主要安全插件如表 1-2 所示。

表 1-2　OSSIM 4 主要插件分布情况

分类	功能	插件名称
1	访问控制	csico-acs、cisco-acs-idm、cisco-asa、f5-firepass
2	防病毒	Avast、gfi security、mcafee、clamav、sophos、panda、netkeeper、

（续表）

分类	功能	插件名称
3	防火墙	cisco-pix、ipfw、m0n0wall、netscreen-igs、motorola-firewall、iptables、fortigate、stonegate、isa-server、sonicwall、m0n0wall、pf、cyberguard、lucent-brick、paloalto、juniper-srx、sidewinder
4	HIDS	Ossec、ossec-single-line
5	IDS	Dragon、juniper-idp、tippingpoint、kismet、osiris
6	蜜罐系统	Glastopng、honeyd
7	负载均衡	Allot、cisco-ace、citrix-netscaler、f5、heartbeat
8	交换机	Cisco、cisco-nexus-nx-os、netgear、nortel、extreme、raslogd、alteonos
9	网络监控	ntop-monitor、p0f、prads、session-monitor、tcptrack-monitor
10	虚拟化	vmware-esxi、vmware-vcenter、vmware-vcenter-sql
11	漏洞扫描	Nessus、Nessus-detector、Nessus-monitor
12	网络服务	Apache、ftp、dhcp、samba、squid、ssh
13	无线接入	aruba-6、extreme-wireless、cisco-wlc、proxim-orinoco

对 OSSIM 插件位置的说明：在安装时系统将支持的插件，全部复制到目录/etc/ossim/agent/plugins/中，如 Nagios 插件的扩展名为“.cfg”的文本文件，可以用任何编辑器修改，在每个插件配置文件中最难理解的当属正则表达式（RegExp）。OSSIM 4 下主要插件如图 1-10 所示，除此以外还包括实现优先级队列的 RabbitMQ 插件和实现自动化部署的 Ansible 插件。

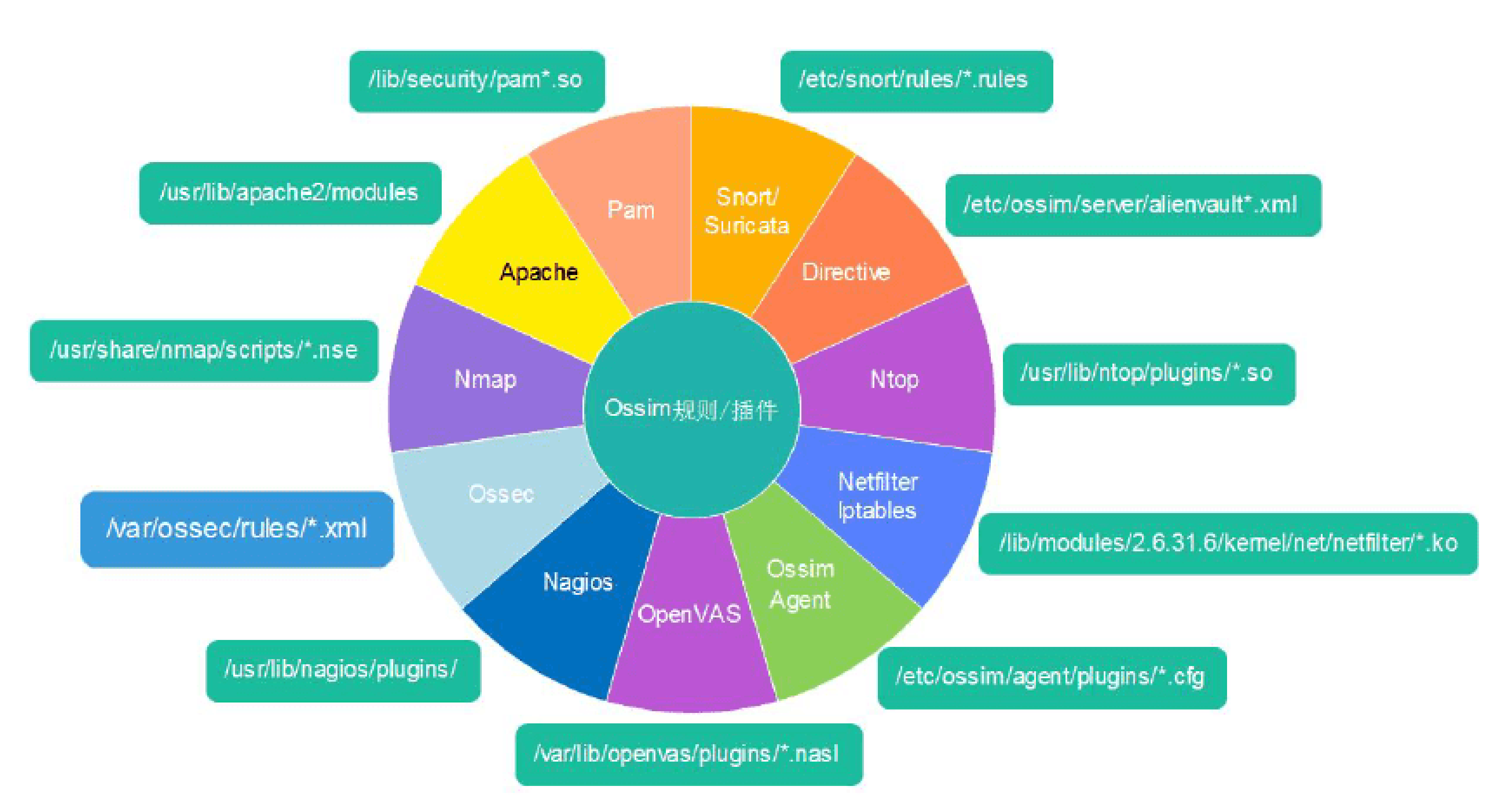


图 1-10　规则与插件路径

在今后安装过程中，选择多少插件系统就在开机时加载多少（插件加载越多越占用内存）。具体查看插件详情，可以访问/etc/ossim/agent/config.cfg 配置文件，而且在系统/etc/ossim/ossim_setup.conf 的[sensor]项中也详细列出了监控插件（monitors）和检测插件（detector）分别包含了哪些内容。在插件这方面，OSSIM 默认提供了上百个插件，涉及 8 个大类，在表 1-2 中仅列出了一小部分，从插件分布和数量来看，OSSIM 系统几乎包含了常用

的插件类型。例如运维设备 Cisco、CheckPoint、F5、Fortigate、Netscreen、Sonicwall、Symantec 等，如果遇到无法识别设备的插件，只能自己编写插件，后续内容会详细讲到。已加载插件如图 1-11，图 1-12 所示。

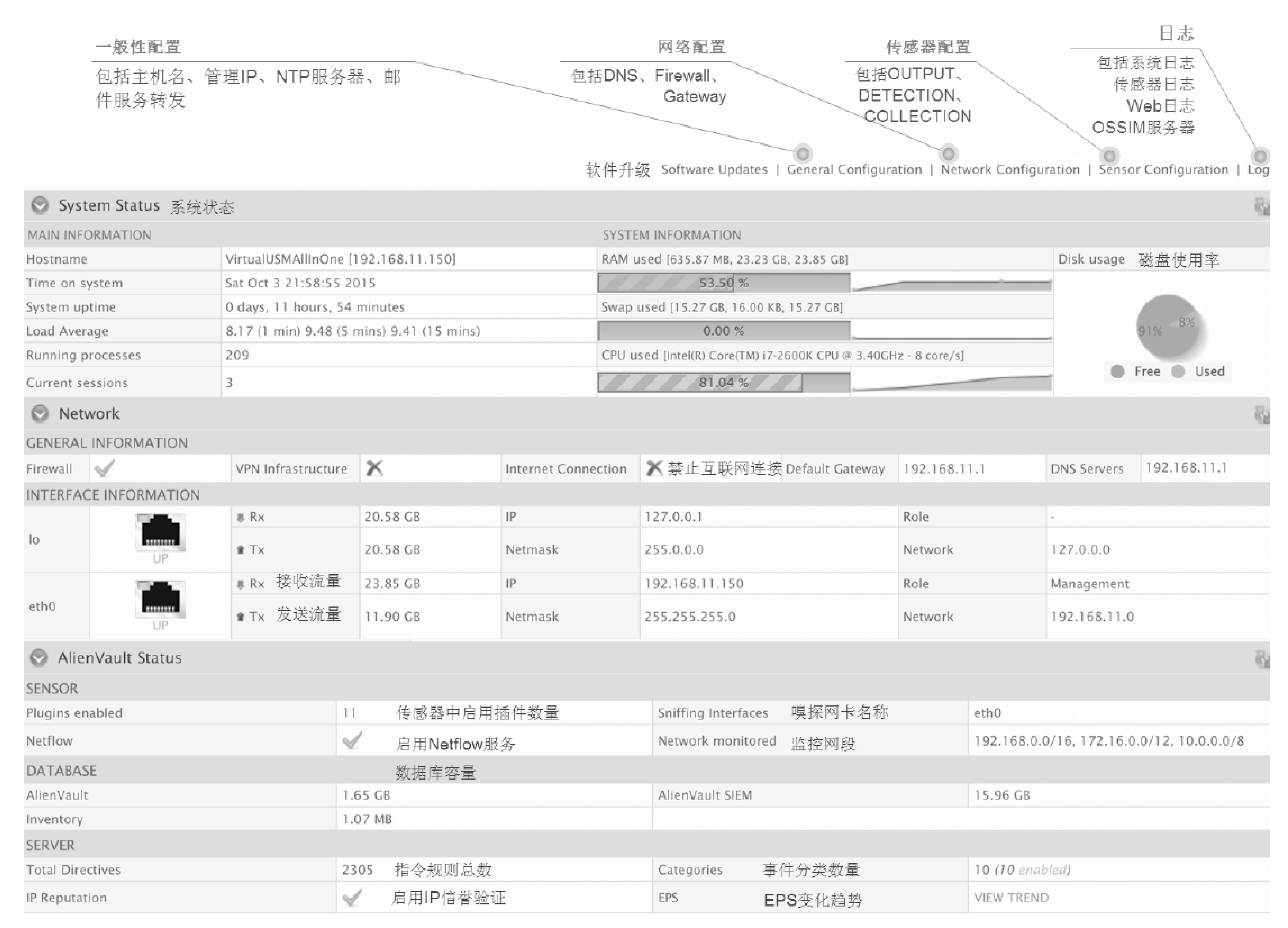


图 1-11　查看传感器启用的插件情况

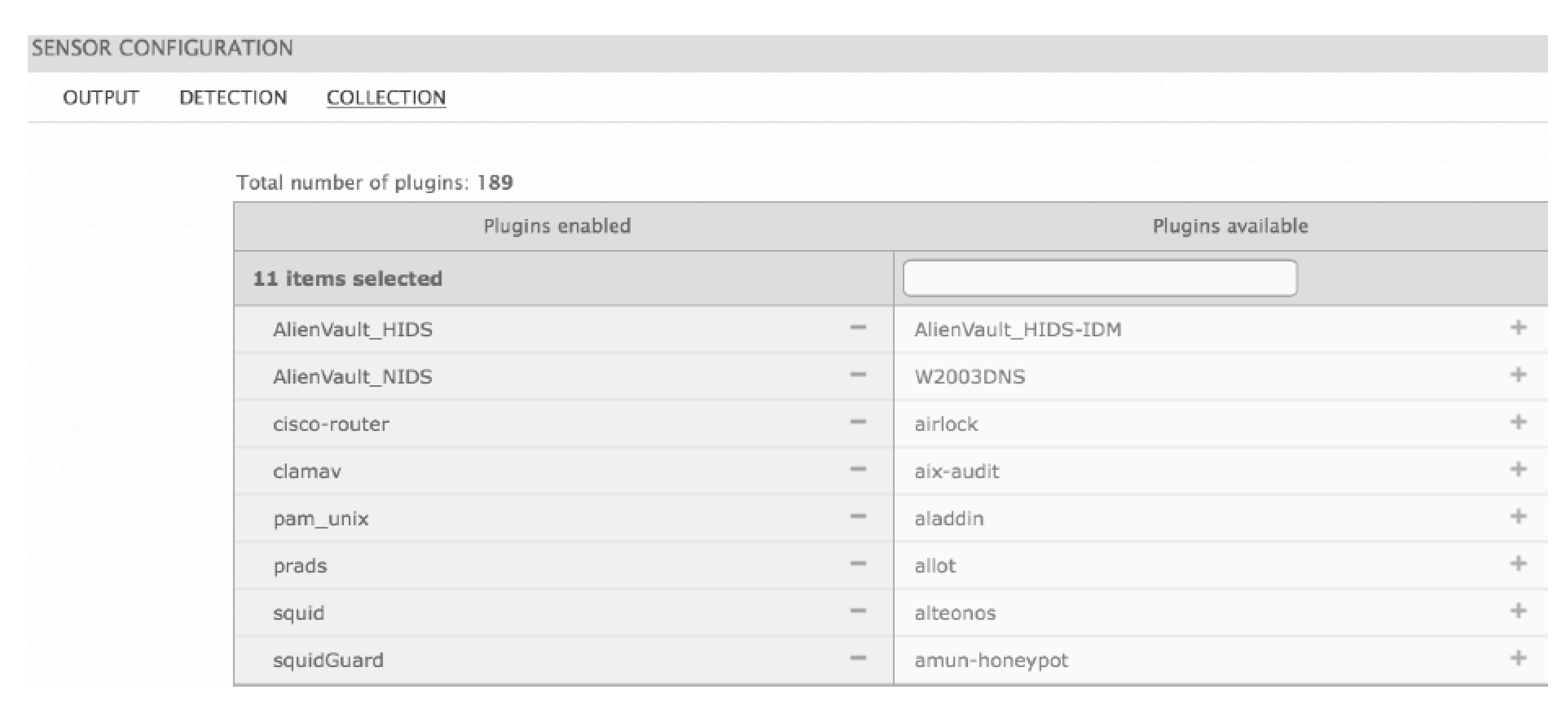


图 1-12　查看加载插件详情

从图中看出，此 Sensor 系统中启用了 11 个插件，如何在 Web 上展示出来呢？如图 1-12

所示总共 184 个插件在 Plugins enabled 中启用了 11 个插件。

从图 1-13 看出这里显示 11 个插件，我们可打开/etc/ossim/ossim_setup.conf 文件查看。

```
[sensor]
detectors=cisco-router, pam_unix, squid, ssh, squidGuard, prads, clamav, sys
log, ossec-single-line, sudo, suricata
ids_rules_flow_control=yes
interfaces=eth0
ip=
monitors=nmap-monitor,ntop-monitor,ossim-monitor
```

图 1-13　从 OSSIM 配置文件中查看插件

1.2.4　检测器（Detector）

OSSIM 下的检测器起到收集资源信息及监听当前网段数据包的作用，主要包括 ntop、prads、suricata、ossec 等，相关内容在后续章节里会详细介绍。

1.2.5　代理（Agent）

代理进程（由于 Agent 采用 Python 语言编写，所以无须编译就能在 Python Shell 环境运行）将运行在多个主机上，负责从安全设备采集相关信息（比如报警日志等），并将采集到的各类信息统一格式，最后将这些数据传至 Server。

从采集方式上看，Agent 属于主动采集，可以形象理解为由 OSSIM Server 安插在各个监控网段的“耳目”，由它们收集数据，并主动推送到 Collector 中，然后 Collector 又连接着消息队列系统、缓存系统及存储系统。

OSSIM 中的这些代理脚本位于/usr/share/alienvault/ossim-agent/目录下，脚本经过加密，以.pyo 为扩展名。例如 OSSIM 代理（ossim-agent）直接读取存储在/var/log/suricata/unified2.alert.1428975051 的日志。

Suricata 的报警输出文件是/var/log/suricata/unified2.alert，这是由/etc/suricata/suricata.yaml 配置文件在 111 行 # alert output for use with Barnyard2 定义，所以 ossim-agent 直接读取该文件就能显示在 SIEM 控制台中。

Agent 的主要功能是接收或抓取 Plugins 发送过来或者生成的日志，经过归一化处理，然后有序地传送到 OSSIM 的 Server，它的功能很复杂，因为它的设计要考虑到如果 Agent 和 Server 之间的网络中断、拥堵、丢包等情况。

在免费版的 OSSIM 系统中，其日志处理大部分情况下不能达到实时，但可以达到准实时（Firm Real-Time），通常会在 Agent 端缓存一段时间才会发送到 Server 端去。Agent 会主动连接两个端口与外界通信，一个是连接 Server 的 40001 端口（在/etc/ossim/agent/config.cfg 配置文件的选项[output-server]中，能看出通信端口设置为 40001），而另一个是连接数据库的 3306 端口。如图 1-14 所示。

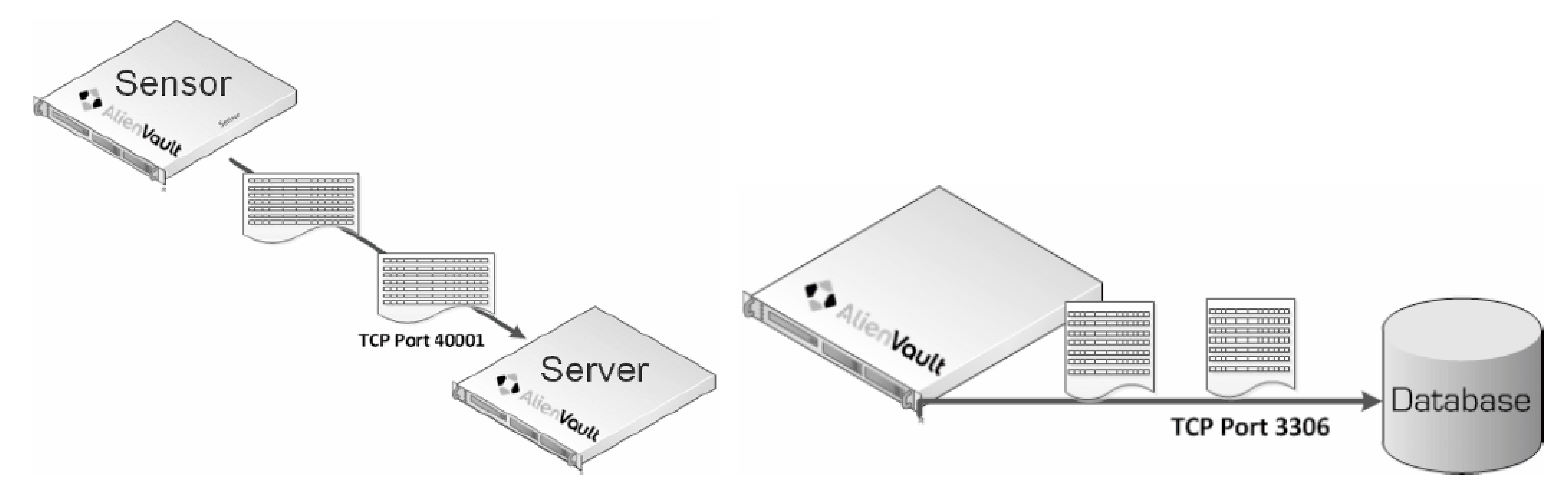


图 1-14　日志归一化处理、收集与存储

原始日志被分成若干段填充到相应的域中，这些字段如下：

- date、sensor、interface、plugin_id、plugin_sid、priority、protocol、src_ip、src_port、dst_ip、dst_port
- username、password、filename、userdata1、userdata2、userdata3、userdata4、userdata5、userdata6、userdata7、userdata8、userdata9

其实 Sensor 的输出数据就是 OSSIM Server 的输入“原料”，我们可在 Web UI 中查看 Sensor 的 output 情况。如图 1-15 所示。

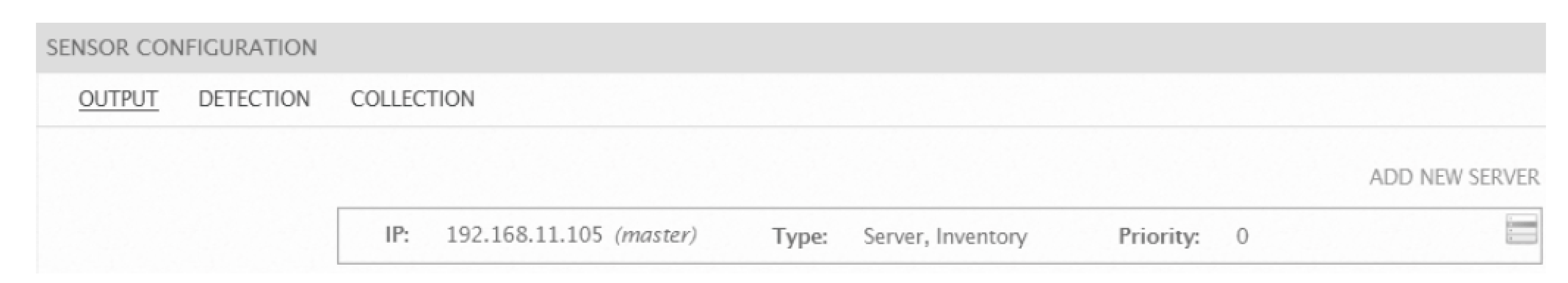


图 1-15　Sensor 将信息输出到 OSSIM server 192.168.11.105 中

1.2.6　报警格式的解码

报警信息的接收过程中，为了应对报警信息格式的变化，OSSIM 采用基于正则表达式的方法对报警信息进行匹配，解析报警事件获取关键信息。正则表达式是一串记录文本规则的代码组合，它的作用可用来进行文本匹配。例如对空白字符、数字字符、中英文字符、IP 和 E-mail 地址等匹配，简单地说它就是一个普通的字符查找串。

例如，下面对一段 Snort 报警信息进行正则表达式的匹配。

```
   5/26-01:02:17.670721 [**] [1:1419:9] SNMP trap udp [**] [Classification:
Attempted
   Information Leak] [Priority: 2] {UDP} 20.20.13.17:162 -> 20.20.20.78:162
   <ids>
        <name>snort</name>
        <method>net_socket</method>
        <regex>^(\d+/\d+-\d+:\d+:\d+).\d+\s+[**] [\d:(\d+):\d+]
\.+[Classification: (\.+)]
```

```
    [Priority: (\d)] {(\.+)}
(\d+.\d+.\d+.\d+)\p*(\d*)->(\d+.\d+.\d+.\d+)\p*(\d*)</regex>
   <order>time,id,classtype,priority,protocol,srcip,srcport,dstip,dstport</order>
    <mapping>snort_classtype,snort_id</mapping>
    </ids>
```

这样通过正则表达式可以提取时间、id、分类、报警级别、协议、源 IP、源端口、目的 IP 和目的端口等信息。接着再将这些字段逐个存储到标准化的表中，分析得到结果，最后通过 Web 界面展现。

1.2.7　OSSIM Agent

Agent 运行在 Sensor 中，负责从各安全设备、安全工具的插件中采集相关信息（比如 IIS 服务日志、Snort 报警日志等），并将采集到的各类信息统一格式，再将这些数据传至 Server，例如将 Snort 系统产生的报警信息收集并存储在 OSSIM Server 中。

OSSIM Agent 中所有脚本采用 Python 编写，相关目录在/etc/ossim/agent/中，代理插件目录在/etc/ossim/agent/plugins/中，配置文件路径为/etc/ossim/agent/config.cfg，OSSIM 系统的代理信息可以通过 Analysis→Detection 下的 HIDS 标签中 Agents 查看。Agent 的结构如图 1-16 所示。

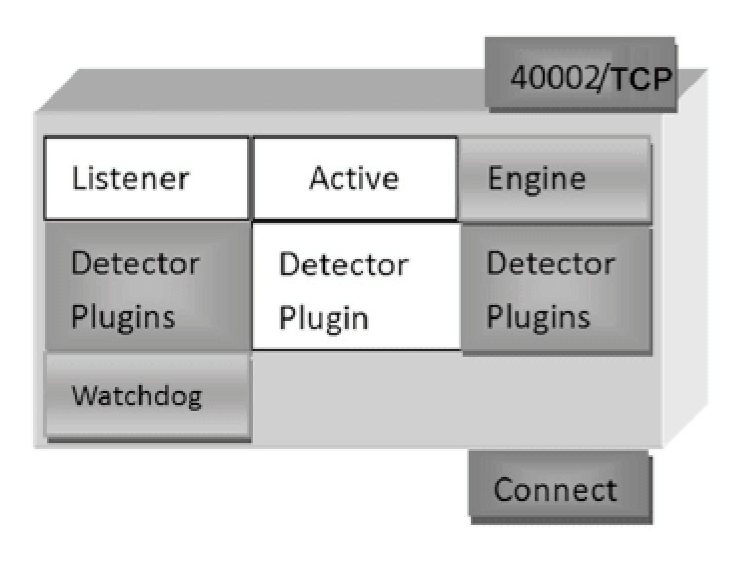


图 1-16　Agent 结构

- 40002/tcp：监听服务器的原始请求。
- Listener：接收服务器连接请求。
- Active：接收服务器输入并且根据请求扫描主机。
- Engine：管理线程，处理监视器请求。
- Detector Plugin：读取日志和进行归一化处理。
- Monitor Plugin：请求监视器数据。
- DB-Connect：连接到本地/远程 OSSIM 数据库。
- Watchdog：监视进程启动/停止进程，检查各插件是否已经开始运行，如遇意外，它会发现并重启相关进程，它自动检查时间为 180 秒，重启进程时间为 3600 秒，其值可以在/etc/ossim/agent/config.cfg 配置文件中修改。

OSSIM 中定义的大部分插件的日志都默认存放在/var/log/syslog 中，所以自定义插件式往往需要修改日志的存放位置，在本书后面的例子中将详细讲解。

表 1-3 分析了目录/etc/ossim/agent/plugin/中主要插件的日志输出情况，通过该表我们很容易了解正则表达式的匹配情况。/etc/ossim/agent/plugings/目录下有 197 个插件，如何能了解每个插件的日志的匹配情况呢？例如，SSH 插件对应的文件为 ssh.cfg，记录的日志文件为/var/log/auth.log，匹配的详情我们通过以下命令实现（在 OSSIM 4.4 之后的版本中已去除了 regexp.py 调试脚本，大家可以到作者博客下载该脚本 http://bjlcg.com:8080/tools/regexp.py，以便完成后续试验）。

表 1-3　插件的日志路径

OSSIM Agent 插件	ID	日志位置
Apache	1501	/var/log/apache2/access.log /var/log/apache2/error.log
Arpwatch	1512	/var/log/ossim/arpwatch-eth0.log
Avast	1567	/var/log/avast.log
bind	1577	/var/log/bind.log
bluecoat	1642	/var/log/bluecoat.log
Cisco-asa	1636	/var/log/cisco-asa.log
Cisco-pix	1514	/var/log/cisco-pix.log
cisco-route	1510	/var/log/syslog
cisco-vpn	1527	/var/log/syslog
Exchange	1603	/var/log/syslog
Extreme-switch	1672	/var/log/extreme-switch.log
F5	1614	/var/log/syslog
fortigate	1554	/var/log/fortigate.log
Fw1ngr60	1504	/var/log/ossim/fw1.log
gfi	1530	/var/log/syslog
heartbeat	1523	/var/log/ha-log
iis	1502	/var/log/iisweb.log
ipfw	1529	/var/log/messages
Iptables	1503	/var/log/syslog
Juniper-vpn	1609	/var/log/juniper-vpn.log
kismet	1596	/var/log/syslog
linuxdhcp	1607	/var/log/ossim/dhcp.log
M0n0wall	1559	/var/log/syslog
mcafee	1571	/var/log/mcafee.log
monit	1687	/var/log/ossim/monit.log
nagios	1525	/var/log/nagios3/Nagios.log
nessus	90003	/var/ossec/logs/archive/archive.log
Nessus-detector	3001	/var/log/ossim/nessus_jobs
netgear	1519	/var/log/syslog
Netscreen-firewall	1522	/var/log/netscreen.log

（续表）

OSSIM Agent 插件	ID	日志位置
nfs	1631	/var/log/syslog
Nortel-switch	1557	/var/log/syslog
openldap	1586	/var/log/openldap/slapd.log
osiris	4001	/var/log/syslog
Ossec-idm	50003	/var/ossec/logs/alerts/alerts.log
Ossec-single-line	7007	
OSSIM-agent	6001	/var/log/ossim/agent.log
P0f	1511	/var/log/ossim/p0f.log
Alienvault-dummy-server	1510	/var/log/ossim/sem.log
prads	1683	/var/log/prads-asset.log
Prads_eth0	1683	/var/log/ossim/prads-eth0.log
postfix	1521	/var/log/mail.log
snare	1518	/var/log/snare.log
Snort_syslog	1001	/var/log/snort/alert
snortunified	1001	/var/log/snort
sophos	1581	/var/log/ossim/sophos.log
suiqd	1553	/var/log/squid/access.log
ssh	4003	/var/log/auth.log
sudo	4005	
Suricata-http	8001	/var/log/suricata/http.log
suricata	1001	/var/log/suricata/
Symantec-ams	1556	/var/log/syslog
Vmware-esxi	1686	/var/log/vmware-esxi.log
vsftpd	1576	/var/log/vsftpd.log
webmin	1580	/var/log/auth.log
websense	19004	/var/log/websense.log

下面看个 Apache 访问日志的例子，如图 1-17 所示。首先在/var/log/apache2/access.log 中的两条日志如下：

```
125.0.0.1 - - [28/Apr/2014:00:54:34 -0400] "GET /ossim/ocsreports/install.php?name=root&pass=KvSNLow
2sL&host=127.0.0.1 HTTP/1.0" 302 279 "-" "Wget/1.11.4"
127.0.0.1 - - [28/Apr/2014:00:54:34 -0400] "GET /ossim/ocsreports/install.php?name=root&pass=KvSNLow
2sL&host=127.0.0.1 HTTP/1.0" 200 2925 "-" "Wget/1.11.4"
```

图 1-17　Apache 日志实例

插件检测格式：

```
./regexp.py    log_filename   regexp  modifier
```

- y：显示不匹配的行。
- v：显示匹配的行。

- q：只显示摘要。

如同在 Linux 中定义别名一样，在 regexp.py 脚本中也定义了别名，区别在于它采用 aliases 定义，详情如图 1-18 所示。

```
aliases['IPV4']="\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
aliases['IPV6_MAP']="::ffff:\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
aliases['MAC']="\w{1,2}:\w{1,2}:\w{1,2}:\w{1,2}:\w{1,2}:\w{1,2}"
aliases['PORT']="\d{1,5}"
aliases['HOSTNAME']="((([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)([a-zA-Z])+)"
aliases['TIME']="\d\d:\d\d:\d\d"
aliases['SYSLOG_DATE']="\w{3}\s+\d{1,2}\s\d\d:\d\d:\d\d"
aliases['SYSLOG_WY_DATE']="\w+\s+\d{1,2}\s\d{4}\s\d\d:\d\d:\d\d"
```

图 1-18　定义别名

为了方便插件内容的定义，插件中使用了经常用到的内容来定义别名，这样做的好处是今后在定义某一插件内容时，可使用已定义的别名代替那个元素，比如用 IPV4 代替 \d{1,3{\.\d{1,3}\.\d{1,3}\d{1,3}的定义。由此可知，插件中定义的 Apache 访问日志的正则表达式为：

```
    regexp=(\IPV4) (\S+) (\S+)
\[(?P<date>(\d\d)\/(\w\w\w)\/(\d\d\d\d):(\d\d):(\d\d):(\d\d)).+"(?P<info>.+)"
(?P<sid>\d+) (\S+)
```

通过插件匹配的详细结果如图 1-19 所示：

```
alienvault:/usr/share/ossim/scripts# /usr/share/ossim/scripts/regexp.py /var/log/apache2/access.log
 /etc/ossim/agent/plugins/apache.cfg v |more_

Multiple regexp mode used, parsing /etc/ossim/agent/plugins/apache.cfg
Matched using 1 - apache-access
127.0.0.1 - - [28/Apr/2014:00:54:34 -0400] "GET /ossim/ocsreports/install.php?name=root&pass=KvSNLow
2sL&host=127.0.0.1 HTTP/1.0" 302 279 "-" "Wget/1.11.4"

[('', '', '', '127.0.0.1', '-', '-', '28/Apr/2014:00:54:34', 'GET /ossim/ocsreports/install.php?name
=root&pass=KvSNLow2sL&host=127.0.0.1 HTTP/1.0', '302', '279', '-', 'Wget/1.11.4')]
Matched using 1 - apache-access
127.0.0.1 - - [28/Apr/2014:00:54:34 -0400] "GET /ossim/ocsreports/install.php?name=root&pass=KvSNLow
2sL&host=127.0.0.1 HTTP/1.0" 200 2925 "-" "Wget/1.11.4"
```

图 1-19　Apache 匹配结果

由此可见，经过处理后的日志内容并没有改变，为了适应 SIEM 事件显示需要，实际日志中各项的排列顺序发生了改变，目的是方便阅读，方便更好地展示在 SIEM 控制台上。再接着看 SSH 的日志，如图 1-20 所示。

```
    #/usr/share/ossim/scripts/regexp.py /etc/ossim/agent/plugins/ssh.cfg
/var/log/auth.log q
```

```
alienvault:/etc/ossim/agent/plugins# /usr/share/ossim/scripts/regexp.py /etc/ossim/agent/plugins/ssh
.cfg /var/log/auth.log  q
/var/log/auth.log
Counted 348 lines.
Matched 1 lines.
alienvault:/etc/ossim/agent/plugins#
```

图 1-20　SSH 匹配结果

又比如：

```
   #/usr/share/ossim/scripts/regexp.py  /var/log/snare2.log
/etc/ossim/agent/plugins/landesk.cfg q
   Multiple regexp mode used, parsing /etc/ossim/agent/plugins/landesk.cfg
   ---------------------------------------------------------------------------
   Rule:  Landesk-sync-job-successed
   Matched 273 times
   Counted 274 lines.
   Matched 273 lines.
```

如果想了解 OSSIM 的 Agent Plugins 所有插件列表，请访问以下网址：

```
   https://forge.fi-ware.eu/scm/viewvc.php/trunk/FI-WARE/Security/Security
Monitoring/ServiceLevelSIEM/config/agent/plugins/?diff_format=s&sortdir=down&p
athrev=46&logsort=rev&limit_changes=0&root=fiware
```

1.2.8 代理与插件的区别

初学者常混淆代理和插件的概念，Sensor（传感器）指软件和硬件的传感器被安装在网络中收集和发送数据，而代理是运行在传感器上，用来收集和发送数据到服务器的一段脚本，一个插件需要了解特定系统的日志格式（例如防火墙、IDS），并通过插件来采集数据。

系统中如果启用插件越多，那么采集到网络中各种数据就越全面。在 Sensor 中加载过多的插件，将会占用 OSSIM Server 的数据库空间，下面的这条经验值需要读者了解，系统中每条事件，约占用 1KB 存储空间，而 1millions 的事件量，大约占 1.5GB 空间。

1.2.9 传感器（Sensor）

传感器（Sensor）俗称探针，用来收集监控网段内各类资产的信息，它工作在网卡的嗅探模式。OSSIM 系统中，把 Agent 和插件构成的一个具有网络行为监控功能的组合称为一个传感器，Sensor 的功能范围主要有：

- 入侵检测（最新版已换成支持多线程的 Suricata）。
- 漏洞扫描（OpenVAS、Nmap）。
- 异常检测（P0f、Prads、ARPWatch 等）。

Arpwatch 主要监视网络中新出现的 MAC 地址，它包含 1 个监视库，名为 arp.dat，在 OSSIM 中位于/var/lib/arpwatch/arp.dat，所以它同样是 AIDE 监控的对象。

OSSIM 具有强大的网络威胁监控，流量监测的功能，但无法将威胁阻断，所以不能将其串联在防火墙链路，最佳方法是作为旁路链接使用，这一点就像部署安全审计设备一样。

在 OSSIM 系统中传感器的状态可在 Configuration→Deployment→Components→Sensors 中查看详情，如图 1-21 所示。每个传感器中加载的插件详情如图 1-22 所示。

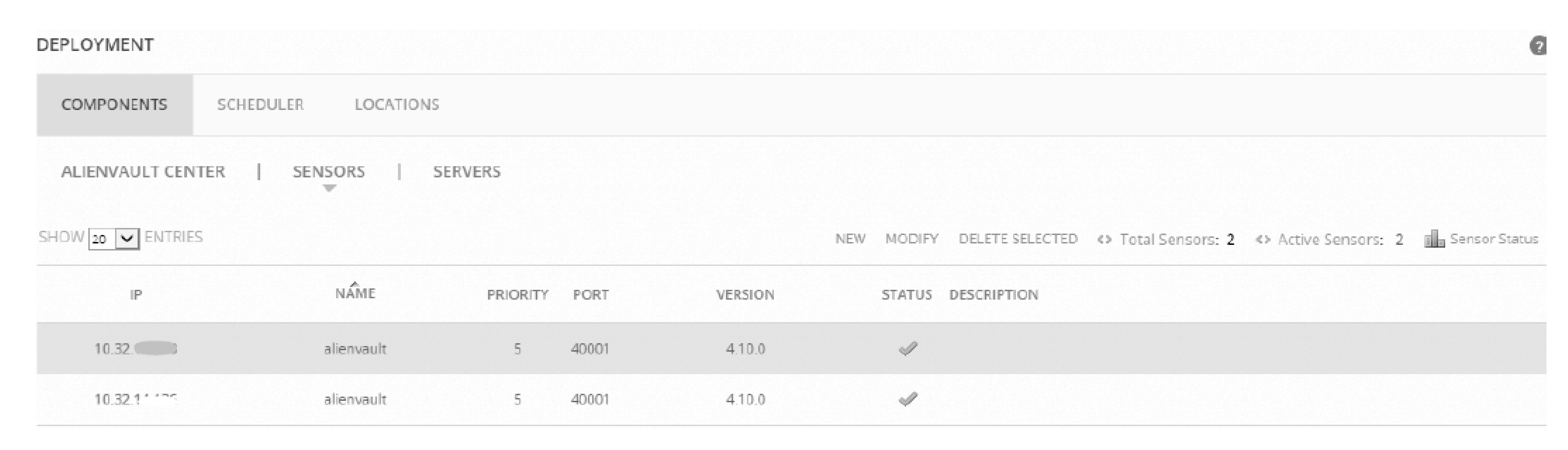


图 1-21　多传感器状态

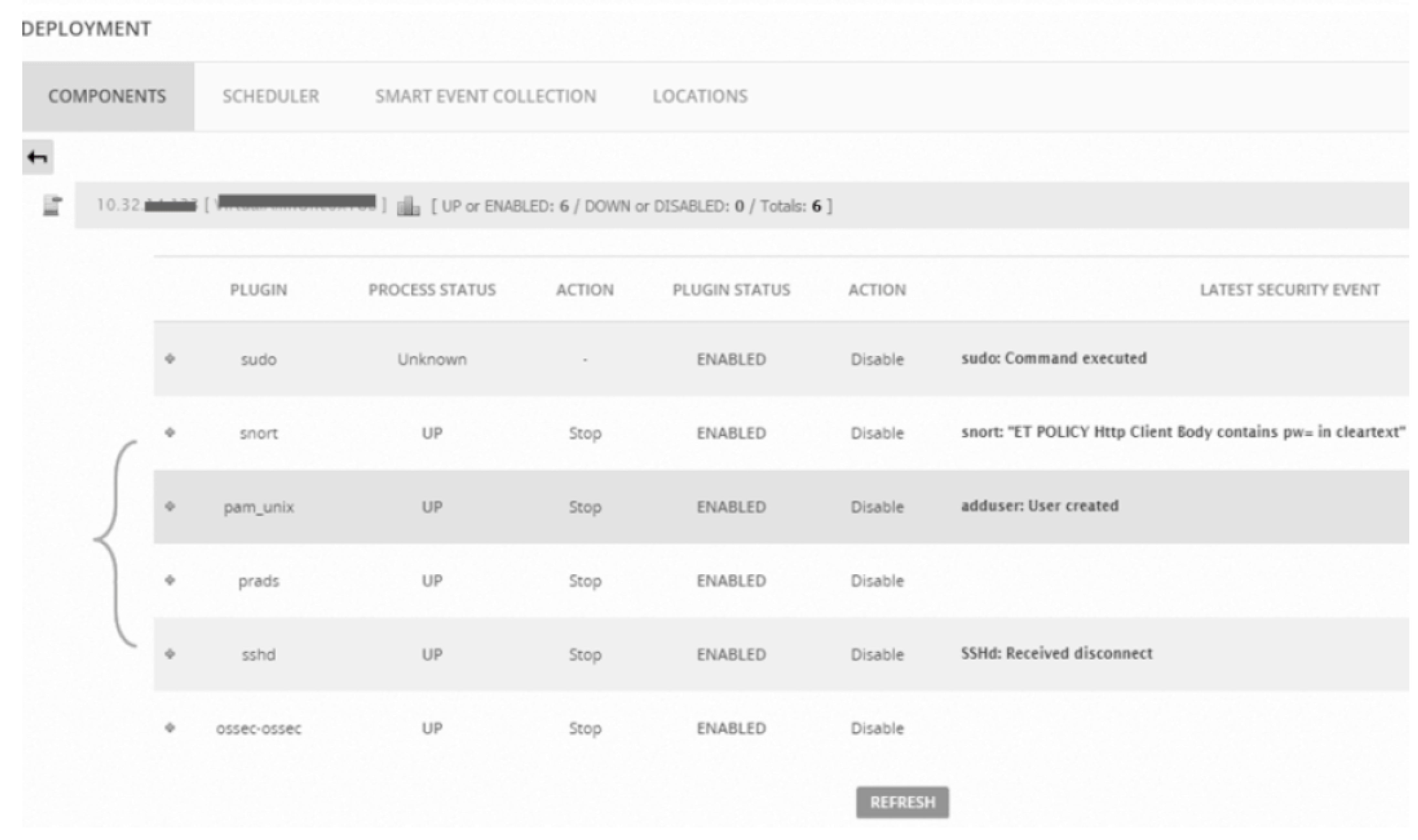


图 1-22　传感器启用插件

在 OSSIM 分布式应用中有多个传感器，这时在图 1-21 中可以查看每个传感器的工作状态，包括 IP 地址、名称、优先级、工作状态等信息。

OSSIM 系统发展到 4.3 版本之后，传感器查询方式发生了变化，路径为 Configuration→Deployment→Alienvault Center→Sensor Configuration→Detection，而且启动程序也发生了变化，由 Suritata 代替了 Snort，如图 1-23 所示。OSSIM Server 默认就启动 ntop、ossec、prads、suricata 这 4 项，Snort 为停止状态。这 5 个检测器的状态无法通过 Web 界面直接进行修改。

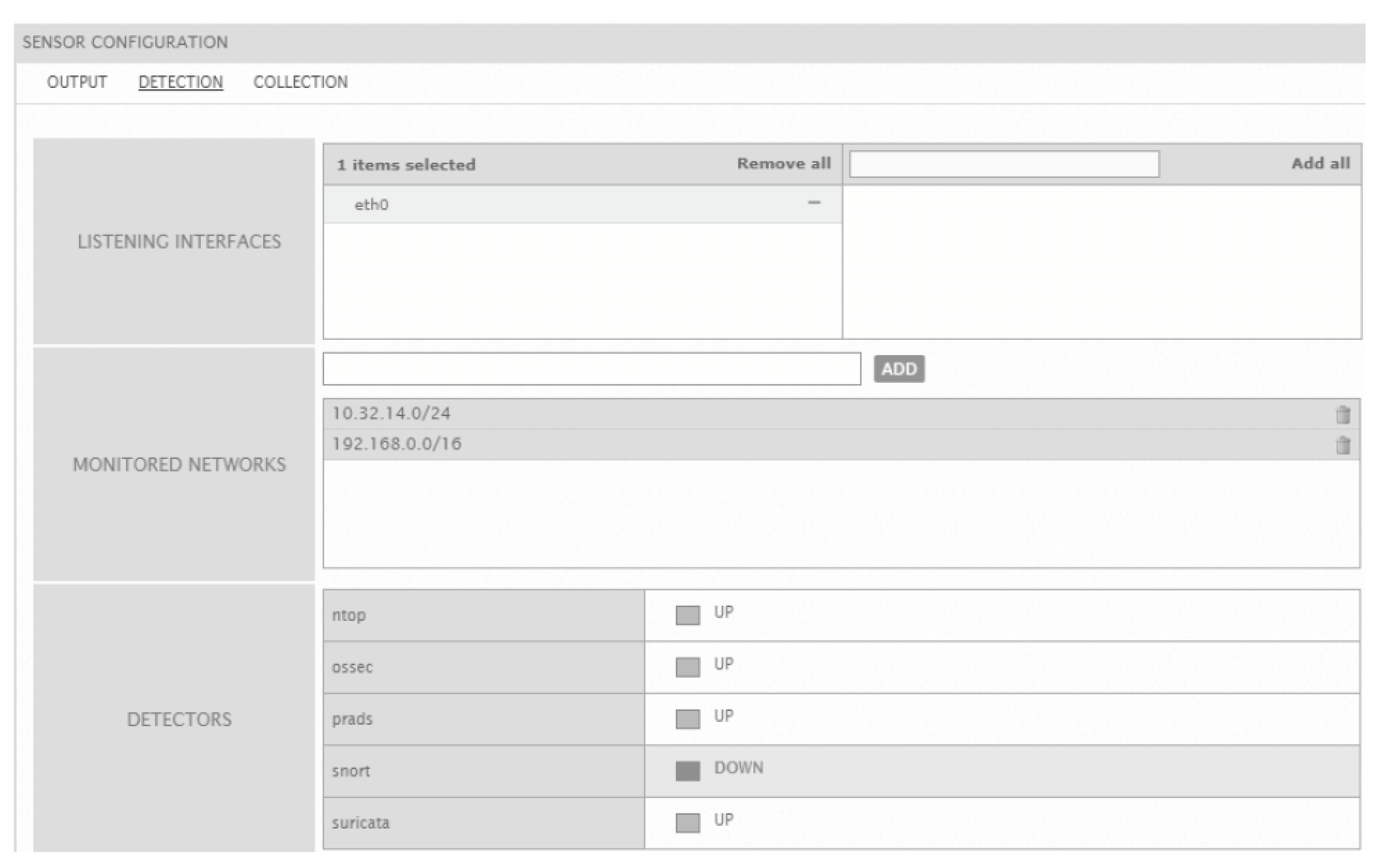


图 1-23　传感器详情

大家如果使用 OSSIM 4.1 系统，查看系统检测插件时，Snort 为启动状态，但 OSSIM 4.2 后的系统 Snort 是关闭状态，代替它的是性能更强大的 Suricata 系统，同一个系统中两者只能任选其一。而 Prads（Passive RealTime Asset Detection System）是 OSSIM 4.2 之后又一款被动实时资产探测系统，它的主要功能就是保存被判断资产特征，如同一个指纹库，保存了各种系统特征，可以识别资产的操作系统类型、由 ARP 发现新增资产，根据开放端口判断打开的网络应用，因为各种设备和服务打开状态并不是一成不变，所以需要用 Prads 来监控变化后的状态。

1.2.10　关联引擎

关联引擎（Server）是 OSSIM 安全集成管理系统的核心部分，它支持分布式运行，负责将 Agents 传送来的归一化安全事件进行关联分析，并对网络资产进行风险评估。其工作流程如图 1-24 所示。

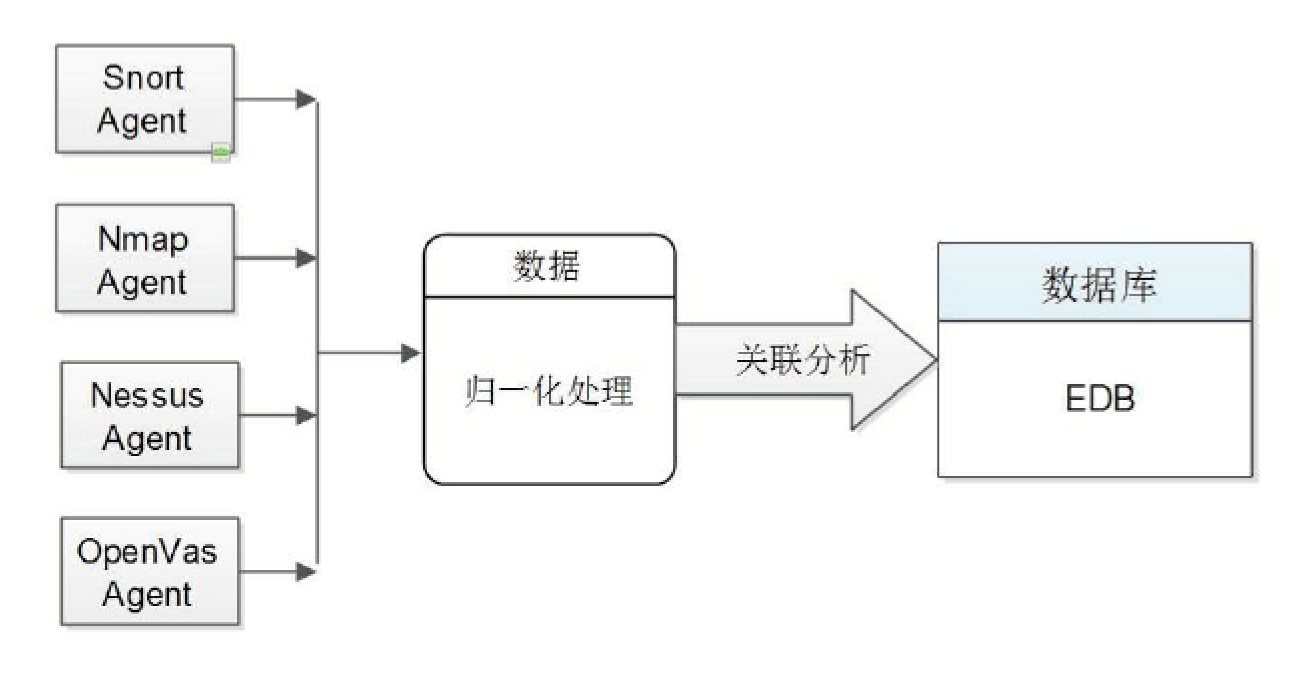


图 1-24　关联引擎的工作流程

OSSIM 服务器的核心组件功能包含：事件关联、风险评估和确定优先次序和身份管理、报警和调度、策略管理、IP 信誉管理等，其配置文件在/etc/ossim/server 目录中，文件分别为：

- alienvault-attacks.xml
- alienvault-bruteforce.xml
- alienvault-dos.xml
- alienvault-malware.xml
- alienvault-network.xml
- alienvault-scan.xml
- alienvault-policy.xml

以上这些文件由开源 OSSIM 免费提供，策略为 84 条，在 USM 中则具有 2000 多条。它们采用 XML 编写易于理解，维护简单。

关联引擎结构如图 1-25 所示，其工作过程由下面 6 个步骤组成：

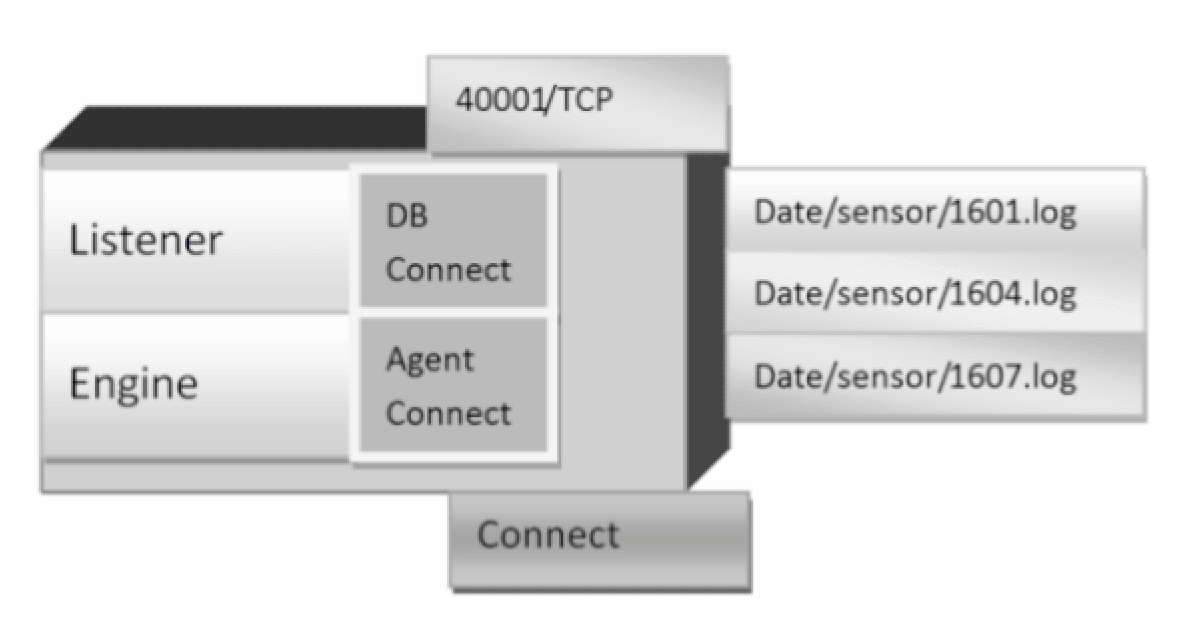


图 1-25　关联引擎的结构

（1）40001/tcp：Server 首先监听 40001/tcp 端口，接收 Agent 连接和 Framework 请求。该端口大小由 OSSIM 系统在配置文件/etc/ossim/ossim_setup.conf 中定义。

我们在 OSSIM 系统中通过以下命令可以清晰查看到其工作端口。

```
#lsof -Pnl +M -i4 |grep ossim-ser
```

（2）Connect：当连接到端口为 40002 指定的 Agent 时，连接到端口为 40001 的其他 Server 对采集事件进行分配和传递。该端口号在/etc/ossim/agent/config.cfg 文件的[output-idm]项配置，不建议更改。

（3）Listener：接收各个 Agent 的连接数据，并细分为 Forwarding Server 连接、Framework 连接。

（4）DB Connect：主要是 OSSIM DB 连接、Snort DB 连接和 OSSEC DB 连接。

（5）Agent Connect：启动 Agent 连接，Forwarding Server 连接。

（6）Engine：事件的授权、关联、分类。

在 OSSIM 系统关联引擎的状态可在 Configuration→Deployment→Components→Servers 中查看，如图 1-26 所示。

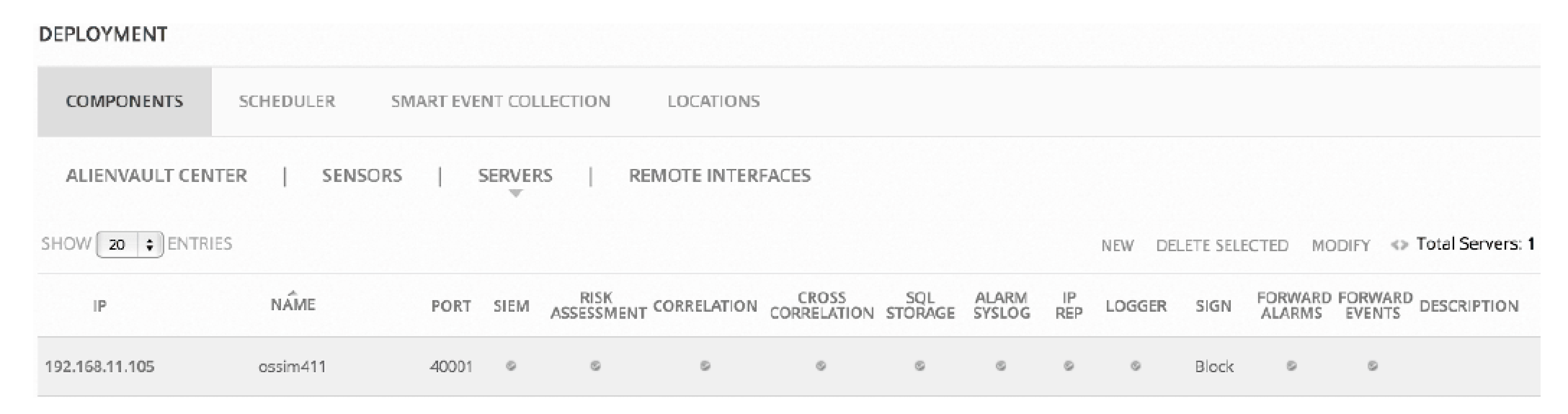


图 1-26　关联引擎模块工作状态

关联引擎启动

OSSIM 系统会启动关联引擎，有时系统调试也需要用到手工启动方式，命令如下：

```
#ossim-server -d -c /etc/ossim/server/config.xml
```

查看 OSSIM Server 版本。

```
#ossim-server -v
```

1.2.11　数据库（Database）

OSSIM 关联引擎（简称 OSSIM Server）将事件关联结果写入数据库。系统用户可通过 Framework（Web 前端控制台）对 Database 进行访问。数据库中 alienvault.event 表是整个系统事件分析和策略调整的信息源。OSSIM 从总体上将其划分为事件数据库（EDB）、知识数据库（KDB）、用户数据库（UDB）。OSSIM 数据库用来记录与安全事件关联及配置等相关的信息，对应于设计阶段的 KDB 和 EDB 的关联事件部分。在 Framework 中使用 ACID/BASE 来作为 Snort 数据库的前端控制台，对应于设计阶段的 EDB。此外 ACL 数据库相关表格可包含在 OSSIM 数据库中，用来记录用户行为，对应于设计阶段的 UDB 库。

OSSIM 数据库分关系型数据库和非关系型数据库。OSSIM 系统默认使用的 MySQL 监听端口是 3306，为增强其处理性能，在 Alienvault USM 中采用 MongoDB 作为非关系型数据库。2013 年将 OSSIM 4.2 发行版中用 Percona_server5.5 替换了原来的 MySQL 5.1，由于使用了 XtraDB 存储引擎，而且对 MySQL 进行了优化和改进，使其在功能和性能上明显提升。在 2015 年最新版本 USM 5.0 中将 Percona_server 升级为功能强大的 5.6.23。OSSIM 主要版本数据库变迁如表 1-4。

表 1-4　OSSIM 主要版本数据库变迁

版本	数据库
OSSIM 2.3	MySQL-server 5.1
OSSIM 3.1	MySQL-server 5.1
OSSIM 4.1	Percona-server-5.5.23
OSSIM 4.2	Percona-server-5.5.25
OSSIM 4.3	Percona-server-5.5.29

（续表）

版本	数据库
OSSIM 4.4	Percona-server-5.5.33
OSSIM 4.5	Percona-server-5.5.33
OSSIM 4.6~4.9	Percona-server-5.5.33-31
OSSIM 4.10~4.15	Percona-server-5.5.33-31.1
OSSIM USM 5.0	Percona-server-5.6.23-72.1
OSSIM USM 5.1	Percona-server-5.6.23-72.1

1.2.12　Web 框架（Framework）

第 1.1 节介绍过 OSSIM 系统涉及 LAMP 环境，由 Perl/Python/PHP 开发工具融合在一起（在/usr/share/ossim/scripts/等路径下有着大量*.py、*.sh 和*.php、*.pl 等脚本文件），它们发挥各自的优势，其中 Web 框架（Framework）控制台，提供用户 Web 页面从而控制系统的运行（例如设置策略），是整个系统的前端，用来实现用户和系统之间的交互。

Framework 可以细分为 2 个部分：Frontend（可视化管理前端）主要采用 PHP 语言编写，它是系统的一组 Web 页面；Frameworkd 是一个守护进程，采用 Python 编写，主要脚本在/usr/share/ossim-framework/ossimframework/目录下，它绑定 OSSIM 的知识库和事件库，监听端口是 40003（在/etc/ossim/ossim_setup.conf 配置文件以及/etc/ossim/server/config.xml 中）可以查看到相关端口定义的信息，同样通过命令（结果如图 1-27 所示）查看：

```
#lsof -Pnl +M -i4|grep ossim-fra
```

```
alienvault:/var/lib/openvas/plugins# lsof -Pnl +M -i 4  |grep ossim-fra
ossim-fra  3209        0    5u  IPv4 606706      0t0  TCP 127.0.0.1:47019->127.0.0.1:3306 (CLOSE_WAI
T)
ossim-fra  3209        0    8u  IPv4   7467      0t0  TCP *:40003 (LISTEN)
ossim-fra  3209        0   10u  IPv4 607235      0t0  TCP 127.0.0.1:47045->127.0.0.1:3306 (ESTABLISH
ED)
ossim-fra  3209        0   11u  IPv4 606708      0t0  TCP 127.0.0.1:47020->127.0.0.1:3306 (CLOSE_WAI
T)
ossim-fra  3209        0   12u  IPv4 607704      0t0  TCP 127.0.0.1:47049->127.0.0.1:3306 (ESTABLISH
ED)
ossim-fra  3209        0   13u  IPv4 602057      0t0  TCP 127.0.0.1:46907->127.0.0.1:3306 (CLOSE_WAI
T)
ossim-fra  3209        0   16u  IPv4   9531      0t0  TCP 192.168.120.77:40003->192.168.120.77:59332
 (ESTABLISHED)
alienvault:/var/lib/openvas/plugins#
```

图 1-27　查看端口

通过以上命令可以清楚查看到服务端口信息，了解这些信息对于我们理解它的工作原理有好处，它负责将 Frontend 收到的用户指令和系统的其他组件相关联，并绘制 Web 图表供前端显示。在 OSSIM 系统中，Framework 安装了 Apache+Php+Adodb 来搭建支持 PHP 的 Web Server，安装 phpgacl 处理用户权限，安装 Mrtg、RRdtool 绘制监控图，安装 ACID/BASE 作为事件的前端控制台。

1.2.13 Ajax 创建交互

OSSIM 的 Web UI 具有很强的可定制性，允许用户更改默认的系统功能，可根据自己喜好，对其显示位置进行个性化定制，比如个性化的主页或仪表盘或者 SIEM 控制台，根据情况增加/删除过滤插件，这些后台通过 jQuery 库实现（/usr/share/ossim/www/js/）jQueryUI 比 JavaScript 更简单，减少了复杂的交互。在 OSSIM 的 Web UI 中利用 Ajax+jQuery 技术实现 UI 创建具有各种定制功能。可以说没有 Ajax+jQuery 就没有 OSSIM UI 中各种丰富的交互式图表的展示。

PHP 开发人员在框架中实现 Ajax（Asynchronous Javascript And XML，异步 JavaScript 和 XML）技术。在框架中使用 Ajax 技术可以改变传统 B/S 应用程序的一些弊端，例如对用户反应不灵敏等。传统的 Web 应用中，用户总是处于提交、等待、响应过程中。而利用 Ajax 技术提供了客户端与服务器异步通信的能力，从根本上让用户从请求、等待、响应的循环中解脱出来，在 OSSIM 中从哪里查看呢？从/usr/share/ossim/www/assets/ajax 下的程序分析得知，OSSIM 前台框架采用了 Ajax 技术。

在 OSSIM 前台框架的描述网页中有很多页面，它们有着相似之处，我们使用 Ajax 加载不同的地方即可，这样提高了页面加载速度，用 Ajax 还可以让页面不用刷新，也能显示不同内容。OSSIM 系统可以说 UI 离不开 Ajax 技术，其整个 UI 可看成是基于 JavaScript 事件驱动，数据由 XML HTTP 获取。详细过程大家可阅读 usr/share/ossim/www/js/prototype.js 源码。

实例：为什么在 OSSIM Web UI 中加载图像时会出现 Loading Widget 的提示?如图 1-28 所示。

图 1-28 图形加载瞬间显示

在 OSSIM 中，通过 Ajax 向页面中加载内容时，由于机器问题会出现“Loading Widget”提示，伴随菊花圈不停旋转，这时页面在加载控件。如果读者具有 PHP 基础，还可以继续分析/usr/share/ossim/www/dashboard/sections/wizard/wizard.php 和/usr/share/ossim/www/dashboard/js/analytics_duo.js。

1.2.14 归一化处理

安全事件关联的基础是安全事件的归一化，也就是将不同数据源的安全事件（表示的格式不同），以及对同一攻击特征产生的攻击类型命名不一致问题（意味着安全事件的名称不一致）进行归一化的过程。首先了解一下日志处理的步骤，如图 1-29 所示。

图 1-29　日志处理步骤

该过程是对报警信息进行格式的统一规范化，将报警信息进行分析与解码，把报警信息中的关键字段提取出来，存储在统一的结构体中，并按不同的类分别归结为一起。

1.2.15　标准的安全事件格式

归一化处理的事件不仅需要统一格式，而且需要专门的属性，我们来看几个典型字段及其说明：

- Alarm：报警名称。
- Event id：安全事件编号。
- Sensor id：发出事件的传感器编号。
- Source IP：src_ip：安全事件源 IP 地址。
- Source Port：src_port：安全事件源端口。
- Type：类型分为两类一类是 detector，另一类是 monitor。
- Signature：触发安全事件的特征值。
- Reliability：安全事件的可信度（描述了一个检测到的攻击是否真的成功可能性，侧面反映了安全事件的严重性质）。

为了更好地学习第 2 章介绍的 SIEM 控制台，我们先了解几个统一格式安全事件的实例。在 Redis 服务器中处理大量的非结构化数据，但最终经过一系列规则检测发出的报警，再经过聚合后产生的聚合报警具有统一格式，并集中存储在 MySQL 数据库中。下面给出典型的记录格式。

（1）Raw Log 典型记录格式如图 1-30 所示。

Event type	Event Detail							
Sensor	Product Type			Category		Sub-Category		
Userdata1	Userdata2	Userdata3	Userdata4	Userdata5	Userdata6	Userdata7	Userdata8	Userdata9
Idm src username	Idm src domain	Idm src hostname	Idm src mac	Idm dst username	Idm dst domain	Idm dst hostname	Idm dst mac	

图 1-30　Raw Log 记录格式

（2）SIEM 事件归一化记录格式如图 1-31、图 1-32 所示。

Normailized Event					
	Date		Event Date	Alienvault Sensor	Interface
	Triggered Signature		Event Type ID	Category	Sub-Category
	Data Source Name		Product Type		Data SourceID
	Source Address	Source Port	Destination Address	Destination Port	Protocol

图 1-31　事件归一化处理格式

SIEM										
	Unique Event ID#		Asets S→D		Priority		Reliability	Risk		
	User name		Userdata1		Userdata2		Userdata3	Userdata4		
	IDM	Src Username@Domain	Src Hostname	Src Mac	Dst Username@Domain		Dst Hostname	Dst Mac		
	Reputation	Source Address	Priority	Reliability	Activity	Destination Address	Priority	Reliability	Activity	

Context				
	Source			
	Hostname	IP	Mac	Context
	Latest Update	Services		Users Info
	Destination			
	Hostname	IP	Mac	Context
	Latest Update	Services		Users Info

图 1-32　SIEM 记录格式

在 OSSIM 中的事件是如何实现存储呢？传感器从各种网络设备和服务器上通过 Rsyslog 服务收集原始日志，存储在 Sensor 所在服务器的硬盘等待处理，当收到日志后，安装在探针服务器上的代理开始工作，利用事先设定好的安全插件开始对日志进行预处理（也就是进行归一化处理），流程如图 1-33 所示。

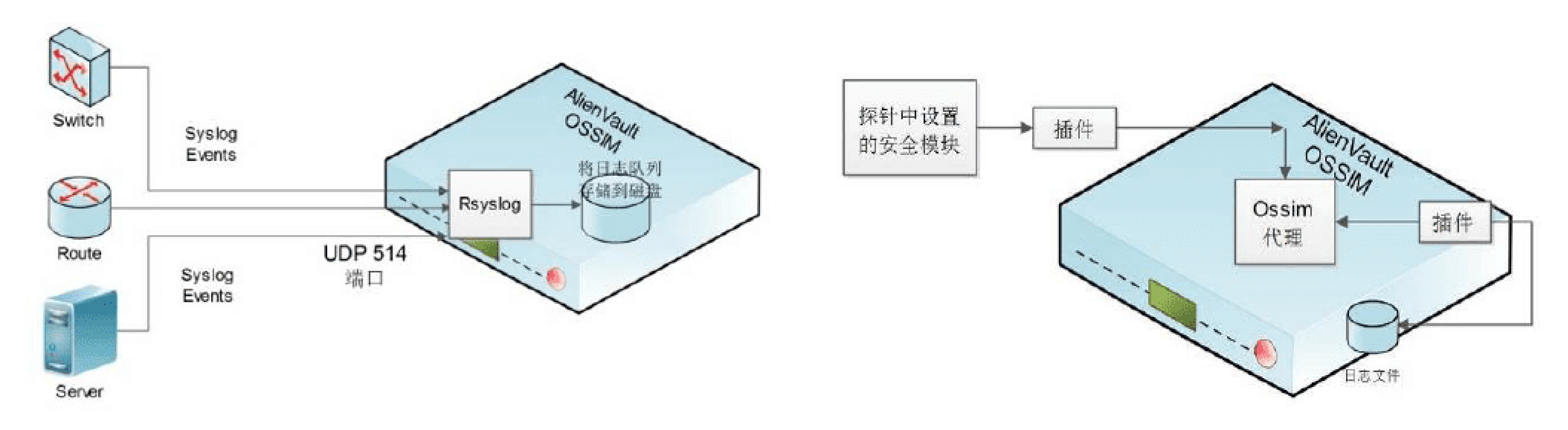


图 1-33　传感器日志采集流程

Agent 将插件收到的日志送往 Server 再进行深度加工，将字段按照类别重新组合成下面的格式（这样就从 RAW Log 变成了归一化处理的日志，归一化处理格式如表 1-5 所示。

表 1-5　归一化处理日志格式

Date	Src_port
Sensor	Dst_ip
Interface	Dst_port
Plugin_id	username
Plugin_sid	password
Prority	filename
Protocol	Userdata1～Userdata5
Src_ip	Userdata6～Userdata9

归一化处理，重要字段含义如下：

- 源和目标地址：在关联分析中属于很重要的内容。
- 源和目标端口：可以分析访问和试图访问的那些服务端口。
- 消息分类：根据用户登录成功、失败或者尝试的消息分类。
- 时间戳：这里包括日志消息在设备上产生的时间和系统接收消息的时间（因为有各种延迟存在，时间不同）。
- 优先级：例如网络设备（交换机）的日志包含了优先级（设备供应商制定）。作为规范化的一部分也需要日志包含优先级。
- 接口：通过哪个网络接口接收到的日志消息。

原始日志是规范化过程的一个重要环节，OSSIM 在归一化处理日志的同时也保留了原始日志，可用于日志归档，提供了一种从规范化事件中提取原始日志的手段。

经过归一化处理的日志，再通过 TCP 3306 端口存储到 MySQL 数据库中，如图 1-34 所示。接着就由关联引擎根据规则、优先级、可靠性等参数进行交叉关联分析，得出风险值并发出各种报警提示信息（详情在后续章节再分析）。

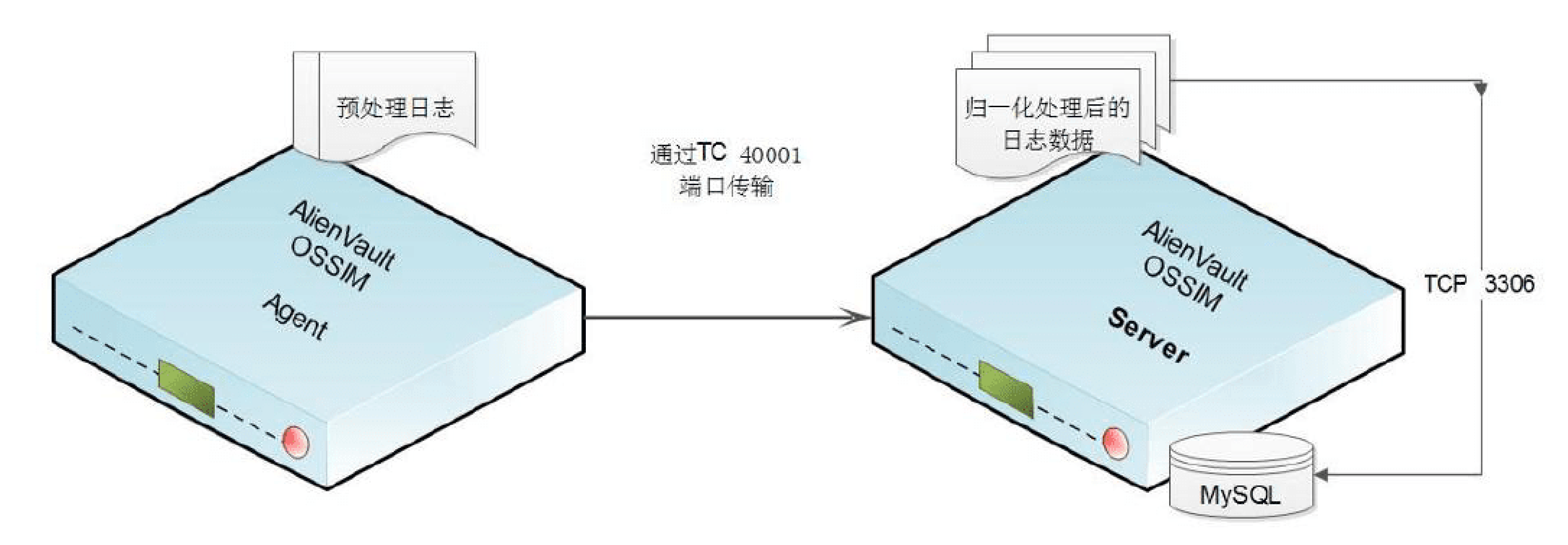


图 1-34　日志存储

接下来，我们再看个实例，下面是一段 Apache 的原始日志，如图 1-35 所示。

Raw Log	AV - Alert - "1379273775" --> RID: "31101"; RL: "5"; RG: "web,accesslog,"; RC: "Web server 400 error code."; USER: "None"; SRCIP: "192.168.11.3"; HOSTNAME: "server"; LOCATION: "/var/log/apache2/access.log"; EVENT: "[INIT]192.168.11.3 - - [16/Sep/2013:03:36:14 +0800] "GET /phpmyadmin HTTP/1.1" 404 1861 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_4) AppleWebKit/536.30.1 (KHTML, like Gecko) Version/6.0.5 Safari/536.30.1"[END]";

图 1-35　原始日志

先经过 OSSIM 系统收集加工后，再通过 Web 前端展现给大家，方便阅读的格式如图 1-36 所示。归一化处理后的事件和原始日志的对比方法我们在后面还会讲解。

Normalized Event

Date	Alienvault Sensor	Interface
2013-09-16 03:36:15 GMT+8:00	server [192.168.11.7]	eth0

Triggered Signature	Event Type ID	Category	Sub-Category
ossec: Web server 400 error code.	31101	Application	Web Not Found

Data Source Name	Product Type	Data Source ID
ossec-accesslog	Operating System	7058

Source Address	Source Port	Destination Address	Destination Port	Protocol
192.168.11.3	0	0.0.0.0	0	TCP

SIEM

Unique Event ID#	Asset S → D	Priority	Reliability	Risk
1e3e11e3-8d09-0800-2765-292315b2db52	2->2	2	1	0

username	userdata1	userdata2
None	Web server 400 error code.	192.168.11.3 - - [16/Sep/2013:03:36:14 +0800] "GET /phpmyadmin HTTP/1.1" 404 1861 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_4) AppleWebKit/536.30.1 (KHTML, like Gecko) Version/6.0.5 Safari/536.30.1"

图 1-36　归一化处理以后的 Apache 访问日志

在图 1-36 所示的例子当中，仅使用了 Userdata1 和 Userdata2，并没有用到 Userdata3～Userdata9，这些是扩展位，主要是为了预留给其他设备或服务使用。

经过归一化处理之后，目标地址会标记成 Host-IP 地址的形式，例如：Host-192-168-0-1。实际上归一化处理这种操作发生在系统采集和存储事件之后，关联和数据分析之前，在 SIEM 工具中把采集过程中数据转换成易读懂的格式，如同图 1-37 显示的那样，采用格式化的数据我们能更容易理解。

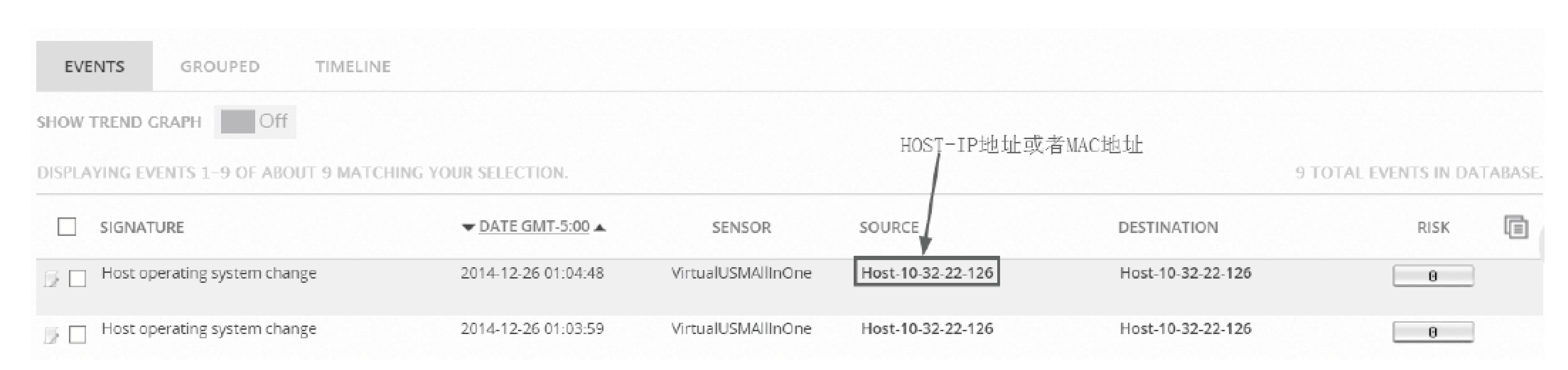


图 1-37　SIEM 控制台下的主机标识形式

1.2.16　OSSIM 服务端口

OSSIM 核心组件包括两部分，一部分是服务器（Server），另一部分是传感器（Sensor）。Server 包括 Server、Web Framework、Database、Identity Management、Vulnerability Management。而 Sensor 包括 Agent、Vulnerability Scanner、Log Collection。它们通信端口如表 1-6、表 1-7 所示。

表 1-6　OSSIM 开放服务器端口分配表

协 议	端 口	进程	作 用
TCP	22	sshd	Alienvault_api 服务器与 Sensor 之间远程通信
TCP	443	apache2	Https-Web UI
TCP	40001	ossim-server	Alienvault-server 服务器进程与 Agent 之间通信端口
TCP	3306	mysqld	Server 和 framework 连接 MySQL 数据库的通信端口
TCP	40002	ossim-server	Alienvault-idm-identity 身份认证进程
TCP	40003	ossim-framework	Alienvault 框架的 Web UI 进程，由 /etc/ossim/server/config.xml 控制
TCP	40004	av-forward	OSSIM 服务器之间的 Log 传送端口（仅在 USM 中）
TCP	40005/40006	machete/mixterd	Alienvault Smart Event Collection Service（仅在 USM 中）
TCP	40011	apache2	API 通信端口，绑定 IP 为 127.0.0.1
UDP	514	rsyslogd	Rsyslog，日志收集服务
TCP	11211	memcached	缓存服务器端口
TCP	5672	rabbitmq	消息服务器（AMQP）
TCP	6379	Redis-server	消息队列存储、加速
TCP	3128	squid	反向代理
TCP	27017	mongod	MongoDB 通信端口（仅在 USM 中）

表 1-7　OSSIM 传感器端口分配表

协议	端口	进程	作用
TCP	22	sshd	SSH 远程安全连接
UDP	555	fprobe	NetFlow 探针
TCP	9390	openvasmd	OpenVAS 管理客户端（进程名为 openvassmd，Manager daemon of the Open Vulnerability Assessment System）
TCP	9391	openvassd	OpenVAS 漏洞扫描进程，The Scanner of the Open Vulnerability Assessment System

（续表）

协议	端口	进程	作用
TCP	4949	Munin-nod	Munin，传感器的监视服务器
UDP	514	rsyslogd	为 Syslog 协议通信使用，作为日志收集服务
UDP	1514	ossec-agentd	OssecServer 和 Agent 之间的通信端口，作为代理管理服务通信端口使用
UDP	1194	openvpn	远程传感器通过 VPN 连接 Server 的通信端口
UDP	12000 及以上端口	fprobe	用于 Netflow 收集 在 OSSIM 系统中文件/etc/nfsen/nfsen.conf 负责定义，分布式环境中多个 Sensor 启用了 Netflow，则端口号依次为 12000、12001、12002 等
TCP	3000	ntop	NTOP 流量监控

了解上述服务端口的作用，对于今后维护 OSSIM 非常有帮助。例如发生 OSSIM Sever 停止运行的故障，如何找原因？当 OSSIM Server 停止运行后，它将不再监听 40001 端口，此时传感器发送回来的数据也就无法收集到，首先查看端口情况。

```
#netstat -lnt |grep 4000
tcp 0 0 0.0.0.0 40003 0.0.0.0:* LISTEN
```

正常时 40001、40002、40003、40004 端口处于监听状态，一旦出现状况后只有 40003 端口在监听。OSSIM 处理流程与通信端口的关系如图 1-38 所示。

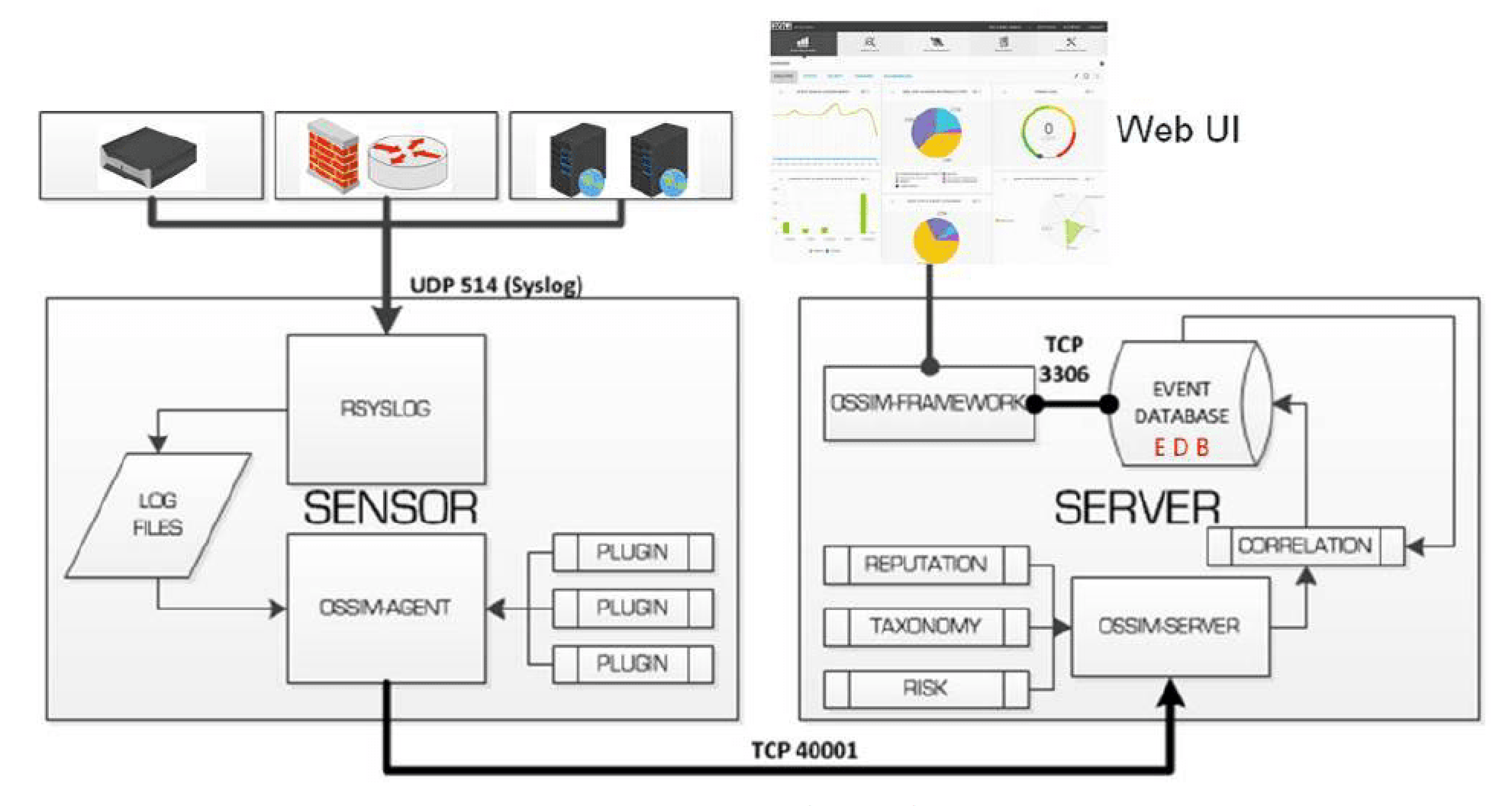


图 1-38 OSSIM 端口通信情况

1.3 基于插件的日志采集

由于现有安全设备产生日志格式不统一，故无法直接进行关联分析，在 OSSIM 系统中采取了基于插件过滤的方式对异构安防设备的日志进行分类采集。

1.3.1　安全事件分类

经过分析发现，这些安防设备的日志之间既有共性也有差异，我们可以按基本类型、子类型和详细类型 3 个层次对安全事件进行分类：

- 信息危害类
- 攻击入侵类
- 恶意代码类
- 信息探测类

1.3.2　采集思路

针对这种安全日志格式及描述内容不统一的问题，在 OSSIM 中采取了基于插件的事件采集代理的收集模式，其基本思路是通过插件来完成日志格式化，在事件采集代理中部署若干个插件，每个插件负责采集某种服务或设备的日志并格式化，再将服务对应端口和插件表示号进行关联与绑定，这样做的优势在于当采集代理接收到设备向监听端口发送的日志后，即可直接调用对应的插件来完成日志格式化任务，且每个插件只能接收绑定端口发送的日志，从而提高了安全事件采集的执行效率。表 1-8 中列举了 OSSIM 系统中数据源与插件 ID 的关系。

表 1-8　OSSIM 主要数据源 ID（plugin_sid）与插件对应描述

Plugin_sid	名称	数据源描述	插件
1001	snort	Snort Rules	snort_syslog.cfg
1002	snort_tag	Snort Tagging	
1003	snort-preprocessors	Snort Dynamic Alert	
1100	snort_spp_portscan	Snort: Portscan preprocessor	
1101	snort_spp_minfrag	Snort: Minfrag preprocessor	
1102	snort_http_decode	Snort: HTTP decoder preprocessor	
1103	snort_spp_defrag	Snort: defragmenter preprocessor	
1104	snort_spp_anomsensor	Snort: SPADE preprocessor	
1105	snort_spp_bo	Snort: Back Orifice preprocessor	
1106	snort_spp_rpc_decode	Snort: RPC preprocessor	
1108	snort_spp_stream3	Snort: stream preprocessor	
1109	snort_spp_telnet	Snort: telnet option decoder preprocessor	

（续表）

Plugin_sid	名称	数据源描述	插件
1110	snort_spp_unidecode	Snort: Unicode decoder preprocessor	snort_syslog.cfg
1111	snort_spp_stream4	Snort: Stream4 preprocessor	
1112	snort_spp_arpspoof	Snort: ARP spoof detector preprocessor	
1114	snort_spp_fnord	Snort: NOP detector preprocessor	
1115	snort_spp_asn1	Snort: ASN.1 validator preprocessor	
1116	snort_snort_decoder	Snort: internal decoder preprocessor	
1117	snort_spp_portscan2	Snort: portscan preprocessor	
1118	snortspp_conversation	Snort: conversation preprocessor	
1119	snort_http_inspect	Snort: http data check preprocessor	
1120	snort_http_inspect_anomalous	Snort: anomalous http server preprocessor	
1121	snort_flow-portscan	Snort: flow decoder preprocessor	
1122	snort_portscan	Snort: portscan decoder preprocessor	
1123	snort_frag3	Snort: fragmentation decoder preprocessor	
1124	snort_smtp	Snort: SMTP preprocessor	
1125	snort_ftp_pp	Snort: FTP preprocessor	
1126	snort_telnet_pp	Snort: telnet preprocessor	
1128	snort_ssh	Snort: SSH preprocessor	
1129	snort_stream5	Snort: TCP preprocessor	
1130	snort_dcerp	Snort: DCE/RPC server preprocessor	
1131	snort_dns	Snort: DNS preprocessor	
1133	snort_dcerpc2	Snort: DCE/RPC server preprocessor v2	
1134	snort_ppm	Snort: ppm preprocessor	
1135	snort_internal	Snort: internal preprocessor	
1138	snort_sensitive_data	Snort: sensitive data preprocessor	
1139	snort_sensitive_data2	Snort: sensitive data preprocessor	
1140	snort_sip	Snort: SIP preprocessor	
1141	snort_imap	Snort: IMAP preprocessor	
1142	snort_pop	Snort: POP preprocessor	
1144	snort_modbus	Snort: Modbus preprocessor	
1501	apache	Apache	apache.cfg
1502	iis	IIS	iis.cfg
1503	iptables	Iptables	iptables.cfg
1507	rrd_threshold	RRD Threshold	rrd.cfg
1510	cisco-router	Cisco router	cisco-router.cfg
1511	p0f	Passive OS fingerprinting tool	p0f.cfg
1514	cisco-pix	Cisco PIX	cisco-pix.cfg
1515	cisco-ids	Cisco Secure IDS	cisco-ids.cfg

（续表）

Plugin_sid	名称	数据源描述	插件
1517	ntsyslog	Windows 系统 syslog 服务	ntsyslog.cfg
1518	snarewindows	Snare 代理 for Windows	snare.cfg
1519	netgear	Netgear 网件路由器/交换机	netgear.cfg
1520	netscreen-manager	Juniper Netscreen Security Manager	netscreen-firewall.cfg
1521	postfix	Postfix mailer	postfix.cfg
1523	heartbeat	HeartbeatLinux 高可用软件	heartbeat.cfg
1524	spamassassin	Spamassassin：反垃圾邮件工具	spamassassin.cfg
1525	nagios	Nagios：主机/服务/网络监控管理系统	nagios.cfg
1526	stonegate	Stonegate 防火墙	stonegate.cfg
1527	cisco-vpn	Cisco VPN	cisco-vpn.cfg
1529	ipfw	FreeBSD ipfw 防火墙	ipfw.cfg
1530	gfi_mailsecurity	GFI MailSecurity 邮件安全网关	gfi.cfg
1551	intrushield	McAfee IntruShield syslog	intrushield.cfg
1553	squid	Squid	squid.cfg
1554	fortigate	Fortinet / Fortigate 防火墙	fortigate.cfg
1555	clamav	Clam AntiVirus 类 UNIX 系统中反病毒软件	clamav.cfg
1556	symantec-ams	Symantec AntiVirus Corporate Edition	symantec.am s.cfg
1557	nortel-switch	北电交换机/路由器	nortel-switch.cfg
1558	sophos	Sophos Antivirus	sophos.cfg
1559	m0n0wall	m0n0wall 防火墙日志	m0n0wall.cfg
1560	pf	pf 防火墙日志	pf.cfg
1561	modsecurity	ModSecurity	modsecurity.cfg
1562	vmware_workstation	Vmware Workstation	vmware-workstation.cfg
1563	optenet antispam	optenet antispam	optenet.cfg
1565	isa-server	Microsoft ISA 防火墙	isa.cfg
1566	aladdin	Aladdin eSafe 内容过滤产品	aladdin.cfg
1567	avast	Avast Antivirus 反病毒软件	avast.cfg
1568	bro-ids	Bro-IDS	bro-ids.cfg
1569	dragon	Enterasys Dragon 凯创入侵检测系统	dragon.cfg
1570	honeyd	Honeyd 虚拟蜜罐系统	honeyd.cfg
1571	mcafee	McAfee Antivirus	mcafee.cfg
1572	sidewinder	Sidewinder firewall (BSD based)	sidewinder.cfg
1573	sonicwall	SonicWALL 防火墙套件	sonicwall.cfg
1574	trendmicro	Trend Micro Messaging Security 趋势科技安全套件	trendmicro.cfg
1575	cyberguard	Cyberguard-SG565 防火墙	cyberguard.cfg

（续表）

Plugin_sid	名称	数据源描述	插件
1576	vsftp	VSFTP 服务	vsftpd.cfg
1577	bind	BIND 服务	bind.cfg
1578	Panda-AS	Panda AdminSecure 熊猫安全卫士	panda-as.cfg
1579	hp-eva	HP EVA 存储管理工具	hp-eva.cfg
1580	webmin	Webmin 服务	webmin.cfg
1581	raslogd	RASlog - Brocade Fabric 博科交换机	raslogd.cfg
1582	serviceguard	HP Service Guard Cluster	serviceguard.cfg
1583	sitescope	HP SiteScope 监控工具	sitescope.cfg
1584	DHCP	Microsoft DHCP 客户端服务日志	dhcp.cfg
1586	openldap	OpenLDAP	openldap.cfg
1587	squidguard	Squid 过滤工具	squidGuard.cfg
1588	lucent-brick	Lucent Brick 阿尔卡特朗讯防火墙	lucent-brick.cfg
1589	radiator	Radiator RADIUS 认证服务器	radiator.cfg
1591	ironport	IRON PORT 思科邮件网关	ironport.cfg
1592	fidelis	FidelisIBM 数据泄露防护系统	fidelis.cfg
1594	cisco-acs-sidb	Cisco-ACS-4-SIDB 思科 AAA 认证系统	cisco-acs.cfg
1595	juniper-netscreen-idp	Juniper NetScreen 入侵防御系统	juniper-idp.cfg
1596	Kismet	Kismet 开源 Wireless IDS	kismet.cfg
1597	cisco-ips	Cisco IPS	cisco-ips.cfg
1598	Symantec	Symantec	symantec-ams.cfg
1603	Exchange	Exchange Server 日志	exchange.cfg
1605	PandaSE	Panda 熊猫安全企业防病毒套件	panda-se.cfg
1607	linuxdhcp	Linux DHCP Service	linuxdhcp.cfg
1608	allot	Allot NetEnforcer 流量控制设备	allot.cfg
1609	Juniper-VPN	Juniper VPN	juniper-vpn.cfg
1610	vyatta	Vyatta events	vyatta.cfg
1611	Siteprotector	Siteprotector IDS	siteprotector.cfg
1612	tippingpoint	3COM Tippingpoint 入侵防御系统	tippingpoint.cfg
1614	f5	F5 负载均衡设备	f5.cfg
1615	paloalto	PaloAlto 防火墙	paloalto.cfg
1616	pureftpd	FTP 服务	pureftpd.cfg
1617	courier	Courier 邮件服务器	courier.cfg
1618	Mcafee-AntiSpam	Mcafee AntiSpam	mcafee-antispam.cfg
1619	SymantecEPM	SymantecEPM: Symantec AV Server	symantec-epm.cfg
1621	Fortiguard	Fortiguard IPS	fortiguard.cfg
1623	Aruba	Aruba Wireless	aruba.cfg
1626	Juniper-SRX	Juniper-SRX Router/Firewall/IDS/IPS	juniper-srx.cfg
1630	bit9	Bit9 可信安全平台	bit9.cfg

（续表）

Plugin_sid	名称	数据源描述	插件
1631	nfs	NFS 服务	nfs.cfg
1632	wuftp	WU-FTP 服务	wuftp.cfg
1633	motorola firewall	Motorola 防火墙	motorola-firewall.cfg
1635	netscreen-igs	Netscreen Device	netscreen-igs.cfg
1636	cisco-asa	Cisco ASA 思科硬件防火墙	cisco-asa.cfg
1640	usbudev	USB Udev 硬件检测	usbudev.cfg
1642	bluecoat	Blue Coat 硬件代理服务器	bluecoat.cfg
1643	stonegate_ips	Stonegate IPS	stonegate_ips.cfg
1646	netkeeper-fw	NetKeeper 防火墙	netkeeper-fw.cfg
1647	netkeeper-nids	NetKeeper 基于网络的 IDS 设备	netkeeper-nids.cfg
1648	dovecot	Dovecot 开源邮件服务器	dovecot.cfg
1649	aix-audit	IBM AIX Audit 服务	aix-audit.cfg
1651	oracle-syslog	ORACLE Syslog	oracle-syslog.cfg
1652	cisco-nexus-nx-os	Cisco Nexus 思科数据中心级交换机	cisco-nexus-nx-os.cfg
1653	cisco-ace	Cisco ACE	cisco-ace.cfg
1654	snare-mssql	MSSQL Server	snare-mssql.cfg
1656	cisco-ips-syslog	Cisco IPS	cisco-ips.cfg
1657	cisco-3030	Cisco VPN	cisco-3030.cfg
1658	vmware-vcenter	VMware Vcenter	vmware-vcenter.cfg
1660	ascenlink-network	Xtera AscenLink 网络负载均衡器	ascenlink.cfg
1663	cisco-wlc	Cisco 2000 系列无线网控制器	cisco-wlc.cfg
1664	axigen	Axigen 邮件服务器	axigen-mail.cfg
1665	tacacs-plus	TACACS+	tacacs-plus.cfg
1666	smbd	Smbd: Samba 服务	smbd.cfg
1667	GlastopfNG	GlastopfNG Web 攻击诱捕蜜罐	glastopng.cfg
1669	dionaea	Dionaea 蜜罐	dionaea.cfg
1670	cisco-asr	Cisco-ASR 1000 系列路由器	cisco-asr.cfg
1672	extreme-switch	Extreme Switch	extreme-switch.cfg
1673	extreme-wireless	Extreme Wireless	extreme-wireless.cfg
1674	f5-firepass	F5 Firepass 远程访问软件	f5-firepass.cfg
1675	drupal-wiki	Drupal PHP 编写的开源内容管理框架	drupal-wiki.cfg
1677	siris-vshell	VanDyke Vshell 远程连接、终端访问工具	siris-vshell.cfg
1678	citrix-netscaler	Citrix NetScaler 负载均衡器	citrix-netscaler.cfg
1679	imperva-securesphere	Imperva SecureSphere WAF 产品	imperva-securesphere.cfg
1680	sendmail	Sendmail	sendmail.cfg
1682	proxim-orinoco	Proxim ORINOCO AP700 无线接入设备	proxim-orinoco.cfg
1683	prads	Passive RealTime Asset Detection System	prads.cfg

（续表）

Plugin_sid	名称	数据源描述	插件
1684	AlteonOS	Alteon 北电 alteon 2424 负载均衡交换机	alteonos.cfg
1685	Suhosin	PHP 保护系统	suhosin.cfg
1686	vmware-esxi	Vmware ESXi server	vmware-esxi.cfg
1687	monit	Monit Plugin	monit.cfg
1688	Storage – StorewizeV7000	IBM storwizeV7000 存储产品	storewize-V7000.cfg
1689	W2003DNS	MS Windows Server 2003 DNS 日志	W2003DNS.cfg
1690	Aruba-6.x	HP Aruba Wireless	aruba-6.cfg
1691	Watchguard	Watchguard Firebox	watchguard.cfg
1802	wmi-system-logger	Wmi-Windows: Agent for Windows	wmi-system-logger.cfg
2004	opennms	OpenNMS 监控系统	opennms-monitor.cfg
2006	tcptrack	tcptrack	tcptrack-monitor.cfg
2008	nmap-monitor	Nmap 扫描工具	nmap-monitor.cfg
2009	ping-monitor	ping-monitor	ping-monitor.cfg
2010	whois	Whois	whois-monitor.cfg
2012	wmi-monitor	wmi-monitor	wmi-monitor.cfg
2013	OCS-Monitor	OCS inventory monitor	ocs-monitor.cfg
3001	nessus	Nessus	nessus.cfg
3002	nmap	Nmap	nmap-monitor.cfg
4001	osiris	Osiris 基于主机的 IDS 系统	osiris.cfg
4003	sshd	SSH	ssh.cfg
4004	pam_unix	Pam	pam_unix.cfg
4005	sudo	Sudo	sudo.cfg
4007	syslog	Syslog 日志记录器	syslog.cfg
6001	ossim-agent	ossim-agent	ossim-agent.cfg
7007	Ossec	Ossec 日志收集	ossec-single-line.cfg
8001	suricata	Suricata HTTP Event	suricata-http.cfg
9555	Fortimail	Fortinet / Fortimail 邮件网关	fortimail.cfg
19004	websense	Websens Web 安全网关	websense.cfg

Sensor 中所列出的每个插件都是一个数据源，通过它来处理日志的标准化，在本书第 7 章日志收集与分析中，分析 Linux 服务和设备的日志，将会用到此表中的内容。

1.4 Agent 事件类型

分布式 OSSIM 系统中，各组件的通信从一级控制中心到二级节点，再到数据采集的 Agent

之间都需要信息交互，它们之间相互传递的信息包括事件、状态、命令及配置文件发布等。OSSIM 系统从不同设备接收的事件日志大致分为普通事件、MAC 事件、操作系统事件和服务事件这 4 种类型，下面分别以这几种类型事件日志格式进行说明。

1.4.1　普通日志举例

```
   event type="detector" date="2012-08-09 12:12:11" plugin_id="4003"
plugin_sid="1" sensor="192.168.150.10" interface="eth0" priority="1"
src_ip="192.168.150.8" dst_ip="192.168.150.8" data="user1" log="Aug  9 12:12:11
ossim-sensor sshd[6567]:  (pam_unix) authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=localhost user=user1"
```

各区域含义如下：

- Type：事件类型，一般有两种类型 Detector 或 Monitor。
- Date：从设备接收日志的时间。
- Sensor：传感器 IP 地址。
- Interface：网络接口。
- plugin_id：称为插件 ID 或安全插件号，表示产生这个事件的插件，也就是可用于区分是哪个 NIDS 或扫描设备产生的事件，这里 plugin_id=4003，代表 SSHd:Secure Shsell daemon。
- plugin_sid：称为 SID 或安全事件号，用于表示安全事件在插件中的事件类型，可用于区分同一探针探测到的不同事件类型，插件的子 ID 在 OSSIM 4.1 中在菜单 Deployment→Collection 下的 DS Groups 选项查询，打开顺序见图 1-39 中的箭头所示 1~4 个步骤。其中数据源插件也是以此插件为基础。其数据源描述如图 1-39 所示。

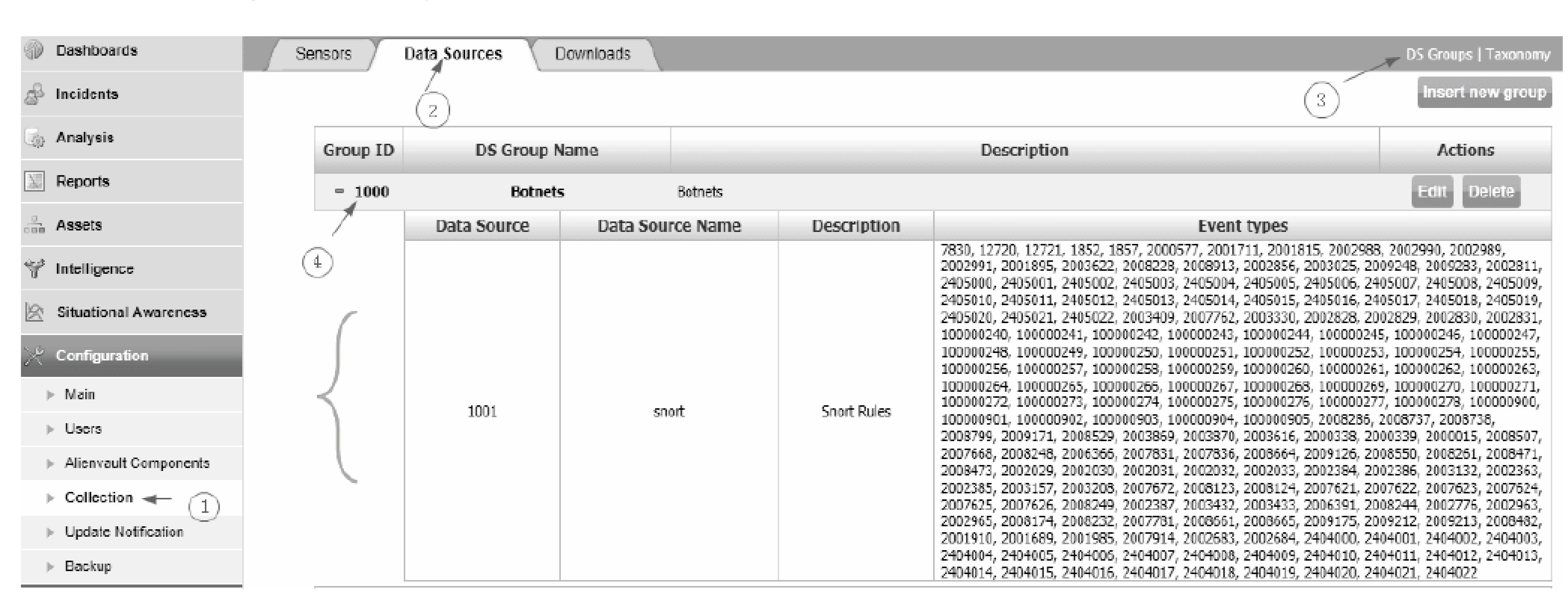


图 1-39　数据源描述信息

1.4.2　plugin_id 一对多关系

每个插件代表对一种数据源采集的定义，数据采集 Agent 通过插件定义的内容分析日志，在此过程中添加事件的 plugin_id、plugin_sid，这样对该事件进行唯一地标识。由于安全设备

往往能产生多种类型安全事件，所以 plugin_id 标识的插件对应多个 plugin_sid 事件。

如果使用了 OSSIM 4.8 以上版本，那么打开方式和上面介绍的有所不同，路径为 Configuration→Threat Intelligence→Data Source，选择 DATA SOURCE GROUPS 按钮，如图 1-40 所示。虽然外观变化了，但一些参数有着共性，其含义需要大家理解。

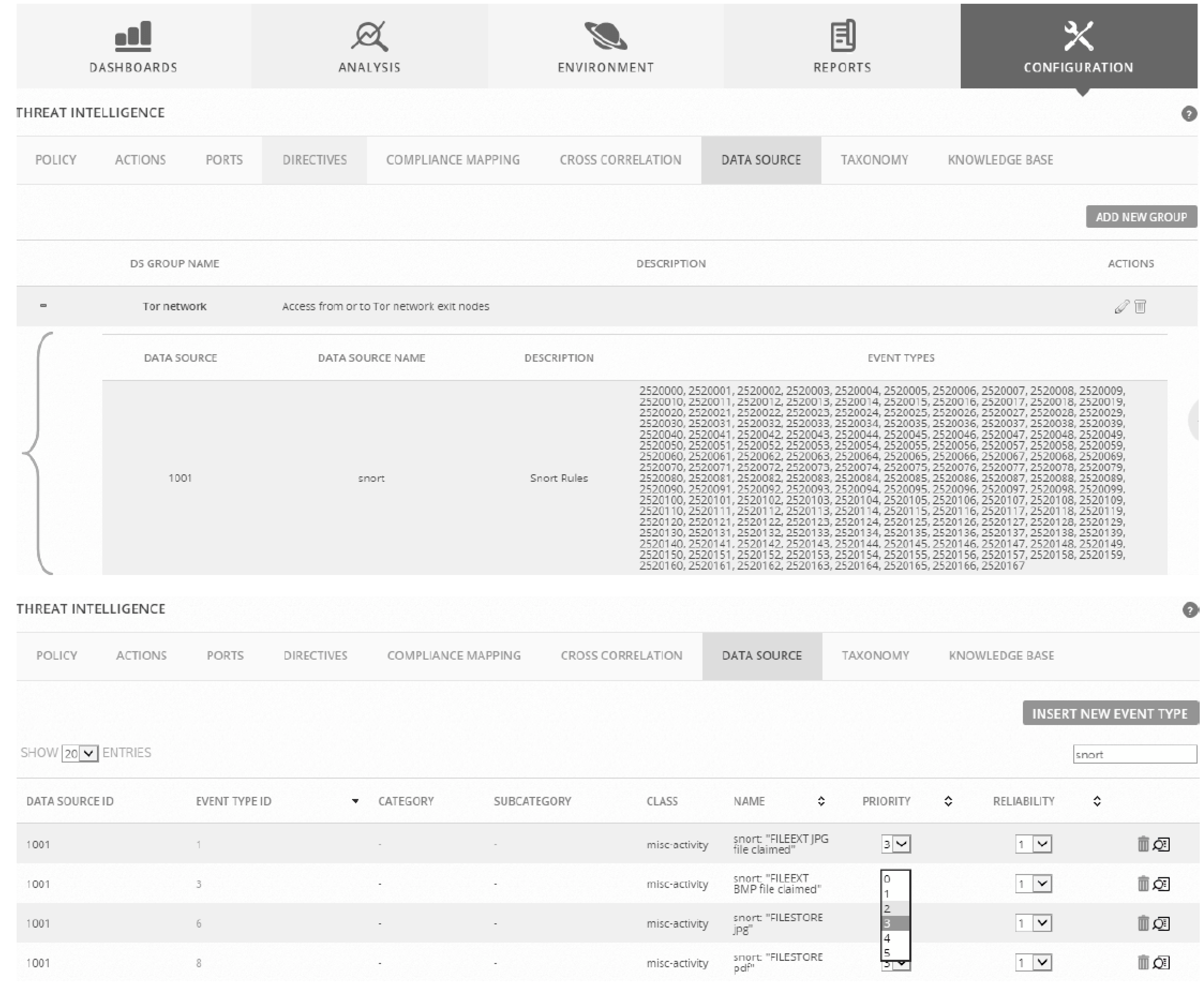


图 1-40　数据源描述

- Priority：优先级（0 为最低，5 为最高）。当修改了插件的优先级之后，risk 会变化从而影响到事件报警，如果需要恢复初始状态，可以在 Web UI 的 Threat Intelligence→Data Source 菜单下单击“Restore Plugins”按钮。
- Timestamp：时间戳。
- Protocol：协议类型，有三种协议类型：TCP、UDP、ICMP。
- Src_ip：源 IP 地址。
- Src_port：源端口。
- Dst_ip：目标 IP 地址。
- Dst_port：目标端口。

- Log：日志内容。

1.4.3 MAC 事件日志举例

```
host-mac-event host="192.168.150.8" interface="eth1" mac="00:24:80:fb:bc" vendor="Intel Corporation" date="2012-03-17 11:30:09" sensor="192.168.150.11" plugin_id="1512" plugin_sid="1" log="ip address: 192.168.150.2 interface: eth0 ethernet address: 0:4:23:88:fb:8b ethernet vendor: Intel Corporation  timestamp: Friday, March 17, 2012 11:30:09 +0100"
```

各区域含义如下：

- host：主机的 MAC 发生改变的 IP 地址。
- mac：十六进制表示的网卡物理地址。
- vendor：网卡厂家。
- sensor：Sensor 的 IP 地址。
- interface：网卡接口。
- date：事件当前日期和时间。

1.4.4 操作系统事件日志举例

```
host-os-event host="192.168.150.8" os="Windows" date="2012-12-20 02:50:13" sensor="192.168.150.10" plugin_id="1511" plugin_sid="1" log="Windows XP" interface="eth0"
```

各区域含义如下：

- host：IP 地址或主机名称。
- os：操作系统。
- sensor：Sensor 的 IP 地址。
- interface：网卡接口。
- date：日期时间。
- plugin_id：操作系统的 pluginID 通常为 1511，它表示 Passive OS Fingerprinting Tool，P0f 工具。

1.4.5 系统服务事件日志举例

```
host-service-event host="192.168.150.77" sensor="192.168.150.10" interface="eth0" port="80" protocol="6" service="www" application="CCO/4.0.3 (Unix) tomcat" date="2012-03-27 07:59:54" plugin_id="1516" plugin_sid="1" log="test_log"
```

各区域含义如下：

- host：地址或主机名称。
- sensor：Sensor 的 IP 地址。
- interface：嗅探网卡。
- port：端口。
- protocol：协议号。
- service：服务种类（例如 WWW、SSH 以及 FTP 等）。
- application：指定应用所对应的服务。
- date：日期时间。
- plugin_id：系统服务 ID 通常显示 1516（这代表 pads 服务，属于网络发现服务，全称为 Passive Asset Detection System）。

在系统报告的事件中，插件 ID 定义插件的类型，我们要在何处才能找到它所有对应的 id 呢？在图 1-41 所示中的查询对话框中输入 id 号就可以方便查到插件用途。

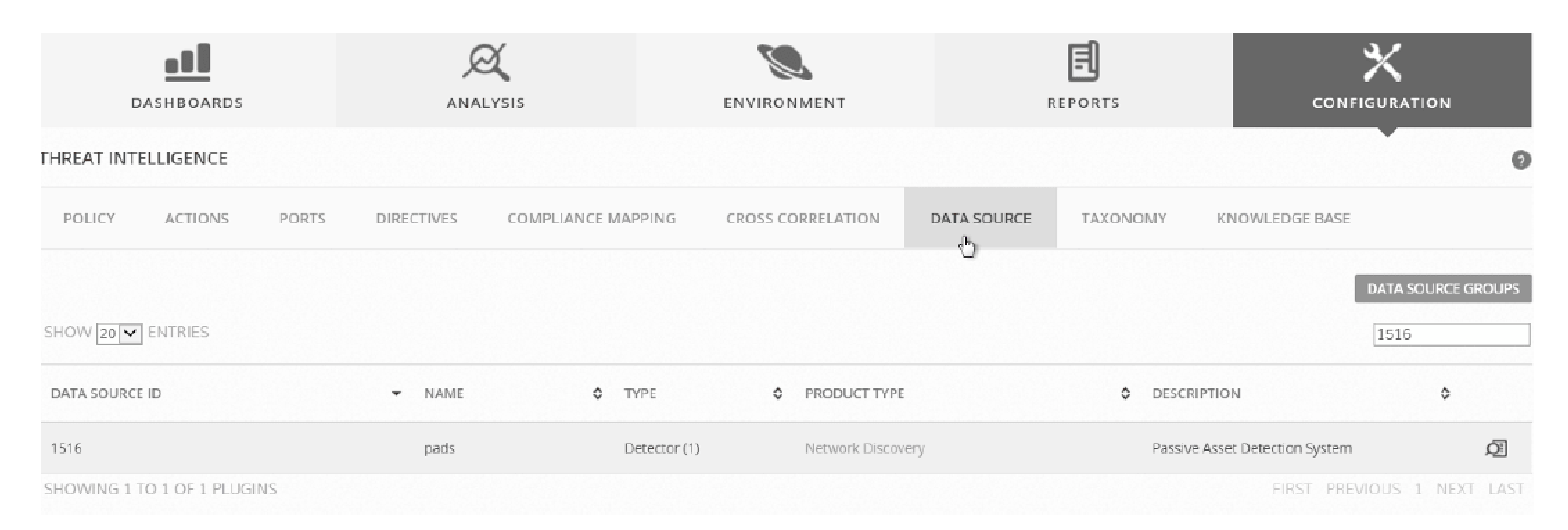


图 1-41　通过插件 ID 号查看插件

另外，在命令行下查询详情，例如 ID:2523，通过以下命令实现。

```
#grep sid:2523 /etc/snort/rules/*
```

1.5 RRDTool 绘图引擎

OSSIM 平台中 Nagios 服务的监控数据来源于 RRDtool，而且其中 Ntop、Munin、Netflow 等重要服务的数据，同样来源于 RRDtool。这些服务共同使用了 RRD Tool 工具来存储 Nagios 性能数据。因为通过 RRD Tool，能够根据先前存储的数据指标创建时序图形，根据这些图形，IT 运维人员能轻易发现某个时间段内，设备的监测信息所发生的变化趋势，可以推断出过去某个时间点，某个特定服务的工作状态如何。例如某服务器 CPU 负载监控截图显示出，在午夜出现了短暂峰值，通过监控图像可发现该现象是偶然还是经常发生，从而推测这种现象背后所隐藏的问题。

1.5.1　背景

RRD Tool 是 OSSIM 系统中的绘图引擎之一，它用于系统绘制各类监控图表，它的前身是 MRTG，但比 MRTG 更灵活。RRD Tool 代表“Round Robin Database Tool”，平时常说的 RRD 就是指 Round Robin Database。

这里所谓的“Round Robin”是一种存储数据方式，它使用固定的空间存储数据，并有一个指针指向最新的位置。可以把用于存储数据的空间看成一个圆，上面有很多刻度，这些刻度所在的位置就代表用于存储数据的地方。

为什么要将 RRD Tool 单独拿出来讲？因为在 OSSIM 很多组件（Nagios、Munin、Ntop、Netflow 等）的图形生成都与它有关。RRD 诞生之前，MRTG 一直是图形化设备利用率数据的行业标准，MRTG 在系统指标收集和存储中引入了 RRD 的概念，

数据源是 OSSIM 里重要的概念，这里为大家介绍的数据源基于 Nagios，从系统负载到网络接口每秒吞吐量，RRD Tool 将这些数据源存储进来，单独的 RRD 文件能够保存用户的大量数据，这些数据可以存储在数据库中。OSSIM 系统的 Nagios、Ntop 以及 Netflow 子系统中，将数据存储在 RRD Tool，用户可以通过命令行工具进行处理。

1.5.2　RRD Tool 与关系数据库的不同

RRDTool 不但具有绘图功能，还具有数据的存储功能，读者可将 RRDTool 理解为数据库，它与关系型数据库相比，区别如下：

（1）用 RRD 的存储格式数据可以重复利用，而其他的数据库只能被动接收数据，RRDTool 可以对收到的整型数据进行计算，如计算两个数据的变化程度并存储结果、将一个 RRD 文件中的数据与另外一个 RRD 文件的数据相加。例如：在 Nagios 可绘制最近 10 分钟的负载，我们可以使用 RRDTool 抓取模式，使用代码：

```
  rrdtool graph /dev/null -start=end-600
'DEF:foo=server1_load.rrd:avg5min:AVERAGE' 'PRINT:foo:AVERAGE:%1f' |tail -n1
```

利用以上脚本进行数据分析，因为数据点原本为 5 分钟的平均值，所以我们还需要计算最近两个数据点的平均值，这样做比较复杂，我们可以利用以上的代码，通过 RRD Tool 的 graph 命令调用 RRD Tool 获取平均值并将其通过标准输出（stdout）输出。

（2）RRD Tool 的 RRD 文件为固定大小，RRD 文件不会像其他关系型数据库文件那样随着时间延长而增大，因此 RRDTool 文件的存储能力有限，不能存放大量数据。

（3）RRD Tool 数据库要求定时获取数据，而关系型数据库则没有这个要求，如果在某个时间间隔内没有收到数据，则会用 UNKN（unknow）代替。

1.5.3　RRD 绘图流程

RRDTool 绘图主要是通过 3 个流程完成，分别是创建 RRD 文件、更新 RRD 数据和绘出

图形。在整个过程中更新 RRD 数据和画图是在指定的时间内重复执行。由于 RRDTool 制图的过程比较复杂，作为普通用户无须了解 RRDTool 的复杂原理和命令就可以通过 OSSIM 平台中配置好的 RRDTool 工具绘制出各种监测图形。

1.6 OSSIM 工作流程

介绍完 OSSIM 系统组件之后，下面讲解其工作流程，主要包括以下 9 个方面的内容：

（1）作为整个系统的安全插件的探测器（Sensor）执行各自的任务，当发现问题时会给予报警；

（2）各探测器的报警信息将被集中采集；

（3）将各个报警记录解析并存入事件数据库（EDB）；

（4）根据设置的策略（Policy）给每个事件赋予一个优先级（Priority），优先级在 OSSIM 系统中为事件管理重要的要素，事件的优先级决定了处理事件顺序和所需要的资源，也就是越紧急的事件优先级越高，需要系统响应时间越短；

（5）对事件进行风险评估，为每个警报计算出风险值；

（6）将设置优先级的事件发送至关联引擎，关联引擎将对事件进行关联，关联引擎就是在入侵检测传感器上报的告警事件的基础上，经过关联分析形成入侵行为判定，并将关联分析结果报送控制台；

（7）对一个或多个事件进行关联分析后，关联引擎生成新的报警记录，将其赋予优先级，并进行风险评估，存入数据库；

（8）用户监视器将根据每个事件产生实时的风险图；

（9）在控制面板中给出最近的关联报警记录，在底层控制台中提供全部的事件记录。

1.7 缓存与消息队列

1.7.1 缓存系统

OSSIM 系统在采用缓冲和缓存框架来提高系统响应的速度。

（1）Memcached

Memcached 是一套高性能分布式的缓存系统，用于动态 Web 应用，以减轻后台数据服务器负载。对于 Memcached 本身细节，这里就不涉及太多了，毕竟这不是本书的重点。下面我们重点看看如何通过 Memcached 来帮助我们提升数据服务的扩展性。要将 Memcached 整合到系统架构中，首先要在应用系统中让 Memcached 有一个准确的定位。仅作为提升数据服务性能的一个 Cache 工具，还是让它与 MySQL 数据库较好地融合在一起成为一个更为高效的数据服务层，更多的操作请参考我的博客《Memcache 与 MySQL 并肩作战》一文。

（2）作为提升系统性能的 Cache 工具

只是系统通过 Memcached 来提升系统性能，作为 Cache 软件，那么更多的是需要通过应用程序来维护 Memcached 中的数据与数据库中数据的同步更新。这时的 Memcached 基本可以理解 MySQL 数据库前端的 Cache 层。这样最大的好处在于可以做到完全不用动数据库相关的架构，所有数据都会写入 MySQL Master 中，包括数据第一次写入时候的 INSERT，同时也包括对已有数据的 UPDATE 和 DELETE。OSSIM 2.3 版中比较适用于需要缓存对象类型少，而需要缓存的数据量又比较大的环境，是一个快速有效的完全针对性能问题的解决方案。稍后再介绍 memcached 性能监控。

（3）Squid Cache

Squid Cache 是目前应用广泛的 Web 缓存服务，可以作为前置缓存服务器来缓存相关的请求数据。OSSIM 中利用在 Apache 前端使用 Squid Cache 作为反向代理，提供文件内容的缓存，以提高系统效率。

1.7.2　消息队列处理

企业日志数量正在以指数级形式高速增长，日志数据的具有海量、多样、异构等特点，基于传统的单一节点混合式安装的 OSSIM 平台（指 OSSIM 4.4 及以下系统），无法满足海量日志分析要求。在 OSSIM 4.4 以后的系统中增加了中间件 RabbitMQ，通过 RabbitMQ 将系统中各组件解除耦合，避免了系统中运行模块的影响（例如 MySQL 的写操作等），这样设计可实现分布式日志分析平台的要求。

在网络出现故障前或故障过程中，各种设备和服务器会发送大量日志到日志服务器，在原先的设计中，这样的日志会直接写入数据库或者/var/log 的某个文件中，在故障状态下的高并发情形下，会对数据库服务器造成巨大的压力，这是在进行日志查询操作时，变得相应延迟加剧，很容易就超过了系统的最大负载。

如图 1-42 所示为早期日志收集的方案，在图中 PHP 脚本从数据库中读取数据的典型流程，此图包括打开数据库连接、运行任意 SQL 语句、读取 SQL 语句找到的结果、关闭数据库连接，最后在 HTML 页面上将所获取的内容显示给用户。

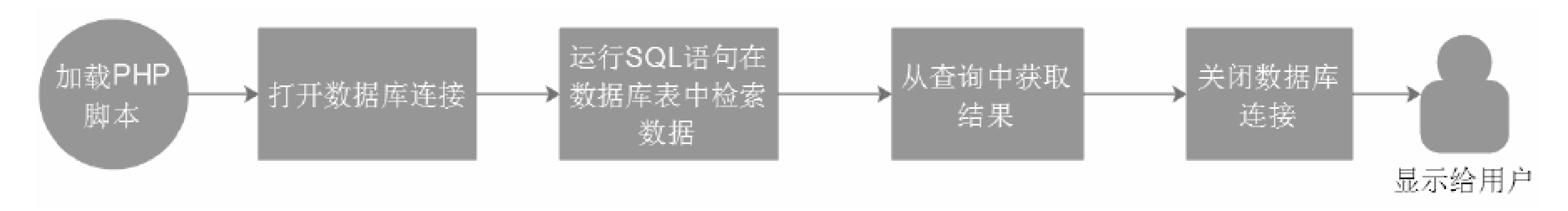


图 1-42　没有缓存情况下从数据库中获取数据流程

上述实例中，每个步骤都存在瓶颈，数据库可能未调整为最佳运行状态，SQL 语句使用的数据表可能未被优化，其中还有很多需要管理员注意的地方。如果没有缓存，用户在每次请求 PHP 脚本时都会碰到问题，每次都会导致性能降低。如果使用缓存来存储 SQL 语句的结果，这种性能的损耗将不复存在。

如何解决这种问题呢，在 OSSIM 中采用异步操作的方式，其核心思想就是消息队列（在 OSSIM 系统中，消息 Message 是由通信的双方所需要传递的日志消息），通过消息队列，将短时间高并发产生的日志消息，存储在消息队列中，从而削平高峰期的并发事务，这样可以有效抵御大量涌入的日志，对单机数据库系统的冲击，也就相对改善了数据库系统性能。

以 OSSIM 中的 SIEM 事件存储来说明问题，设备将日志发送到 Sensor，经过归一化处理之后发往 OSSIM Server，关联引擎对其继续加工，会同时将数个任务收集事件写到数据库，因此数据库写入量很大，当同时查询时并发操作将严重占用有限的 I/O 通道。

由于关联引擎的业务逻辑的处理比较复杂，往往 MySQL 数据库的写操作量更大，所以在 OSSIM 4.0 之前系统中没有采用消息队列时，大负荷时往往系统响应迟缓。

但 OSSIM 4.4 之后采用消息队列的思想，重构了系统之后，加入了消息队列服务器，结构如图 1-43 所示。

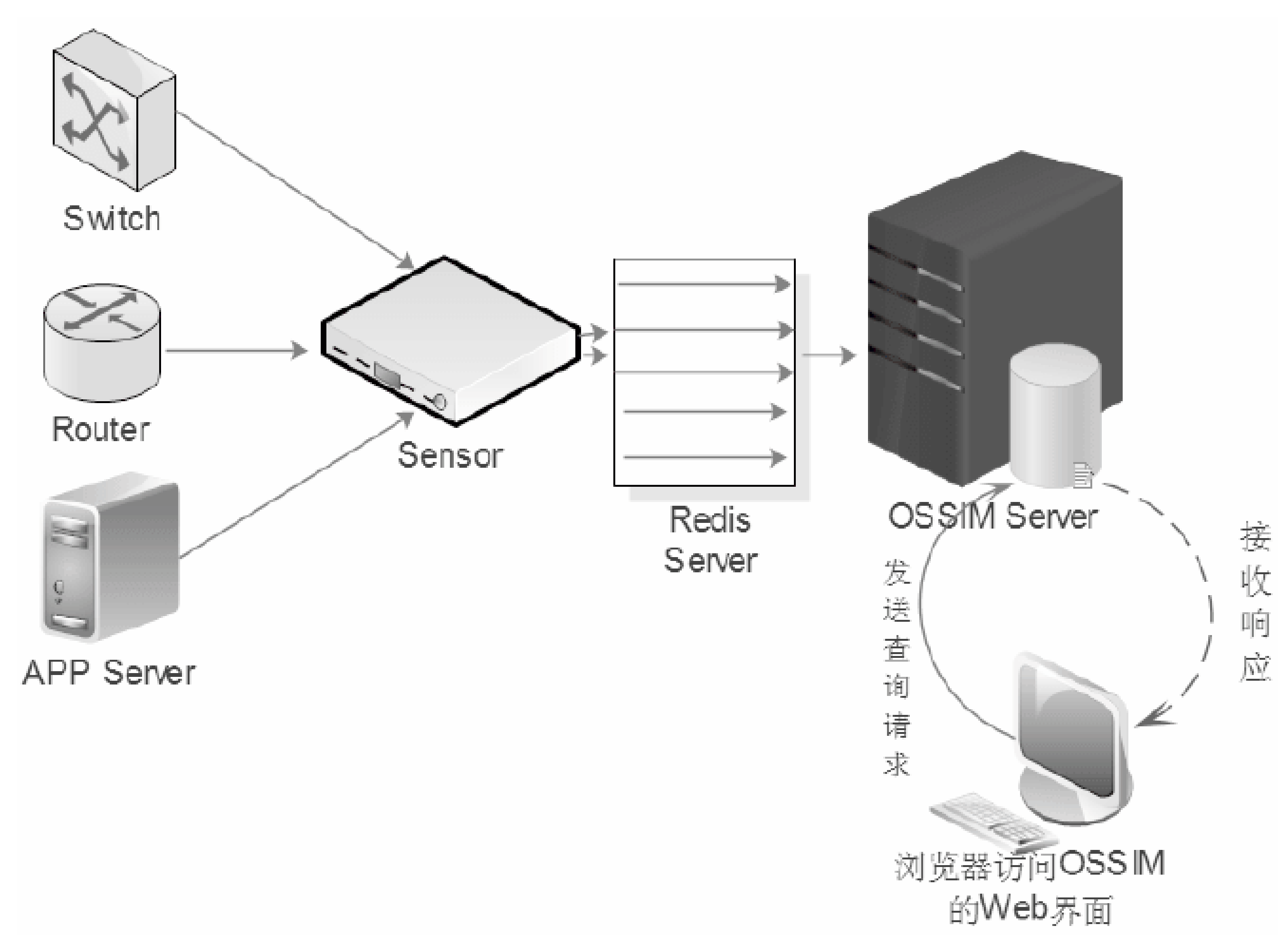


图 1-43　引入消息队列机制

消息队列服务器中有一个进程单独对消息队列进行处理，首先判断消息队列中是否有待处理的消息，如果有的话，那么将其取出，并进行相应地处理。消息队列处理流程如图 1-44 所示，就这样通过消息队列将高并发用户请求进行异步操作，然后逐一对消息队列进行出队同步操作，也避免了并发控制的难题。

图 1-44　消息队列

消息队列只是解决并发问题的其中一种方式，在 OSSIM 中往往需要结合多种不同的技术方式来共同解决，比如负载均衡、反向代理等方案。这里，虽然以异常日志为例，但是“麻雀虽小五脏俱全”，日志写入文件的高并发操作也同样适用于数据库的高并发，所以研究该案例具有一定指导意义。

1.7.3　RabbitMQ

RabbitMQ 是实现 AMQP（高级消息队列协议）消息中间件的一种，它是消息中间件的开发标准，与平台无关，现用于 OSSIM 分布式系统中存储转发消息，其工作原理如图 1-45 所示。

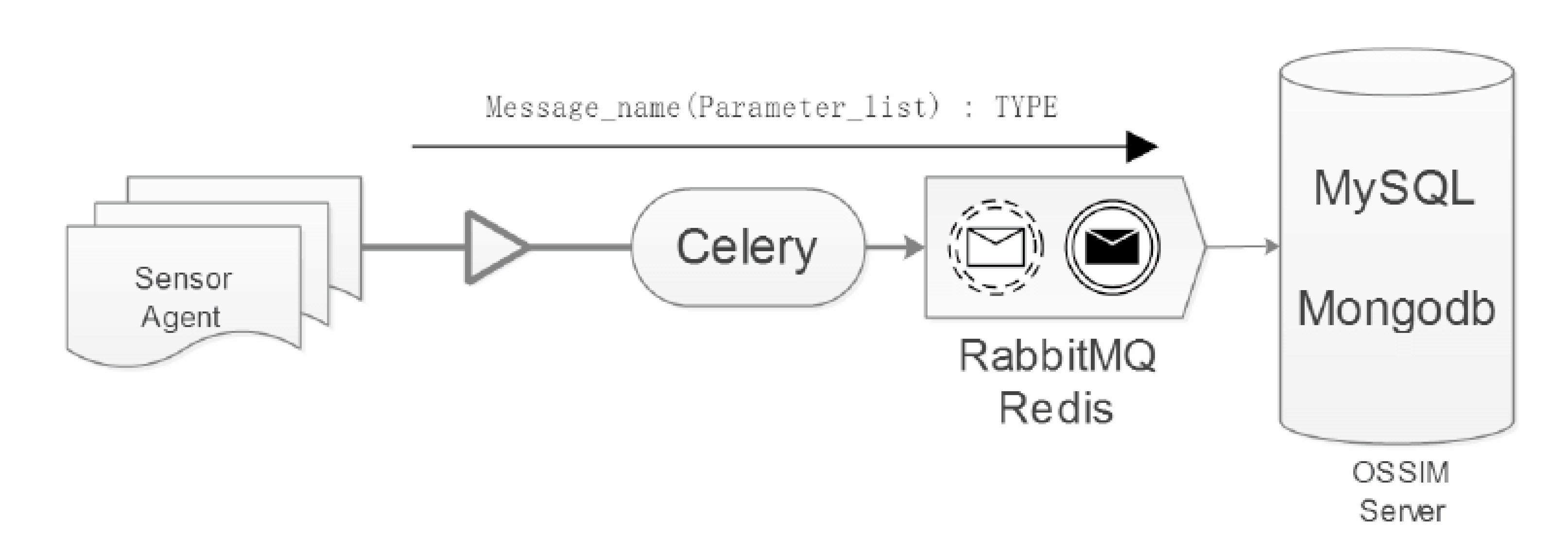


图 1-45　OSSIM 消息队列原理

为什么要使用 RabbitMQ？在高并发情况下，RabbitMQ 在处理发送和接收请求时响应非常快速，而且对其性能调优后，系统响应速度还会有所提高。RabbitMQ 使用 Erlang 编写的 AMQP 服务器，而 AMQP 基于客户端/代理模式。

在大数据日志分析应用中，我们常将 OSSIM 做成集群模式，因为这样可以使用 RabbitMQ 的集群功能。另外，RabbitMQ 的集群中有两种节点：内存节点与磁盘节点。对于每个 RabbitMQ 节点，根据日志种类建立相应的队列，并且根据日志种类的名称建立 exchange 的 key 值（后面再介绍）。RabbitMQ 通信端口为 5672。

OSSIM 中我们通过以下命令查看端口。

```
#netstat -na |grep 5672
tcp 0  0  127.0.0.1:52667    127.0.0.1:5672   ESTABLISHED
```

RabbitMQ 环境变量的配置文件位于/etc/rabbitmq/rabbitmq-env.conf，使用时需要查询 RabbitMQ 状态命令，方法如下:

```
#rabbitmq-server alienvault    \\这里的 alienvault 代表 OSSIM Server 的主机名称
```

此时，能显示 rabbitMQ 的端口、节点名称和主目录（/var/lib/rabbitmq）。

永远不要用 kill 命令直接停止 RabbitMQ 服务器，而应该采用 rabbitmqctl 命令，这样可以将数据同步保存到磁盘，然后关闭服务:

```
#rabbitmqctl stop
```

1.7.4　选择 Key/Value 存储

早期 OSSIM 版本中依然采用 MySQL+Memcached 的架构存储数据，由于存在性能和一致性问题，在 OSSIM 4.4 后续版本中引入 Redis 服务器，有读者会问为什么选用 Key/Value 存储？从 SIEM 收集信息的角度出发，无论日志还是安全事件都属于非结构化数据，在日志关联分析时，非结构化的数据量会大大超过结构化数据，这些数据之间存在关联，如果将这些数据直接存入数据库，那么对数据库访问频率将增加，由于单个表行数的猛增，对表的访问行数成比例增加，在多个表之间的数据调用将会产生大量的计算，足以将任何一个数据分析系统压垮。

在分析 OSSIM 系统各集成开源工具中发现系统没有进行持久化，如果 RabbitMQ 服务器重启会导致消息丢失，所以采用 Redis Server 来解决这些问题，Redis 是一个 Key/Value 的 NoSQL 数据库，Redis 作为缓存服务器，数据存储在内存，所以速度比 MySQL 要快得多，尤其在 OSSIM 的 Web UI 中很多的排行榜应用，取出 TOP 10 的操作：包括各种仪表盘、计算器应用、SIEM 控制台的 Uniq 操作、获取某段时间所有数据的列表、取最新的 TOP N 操作，以及在分布式日志收集系统中消息队列的处理，包括报表输出中产生 Top 10 Attacker Host、Top 10 Used Ports、Top 15 Alarms，这些都要用到 Redis 服务。如图 1-46 是 Redis 在 OSSIM 系统中的典型应用。

图 1-46　Redis 在 OSSIM 系统中的典型应用

1.7.5　OSSIM 下操作 Redis

下面列举了常见的有关 Redis 的操作实例。

（1）查询 Redis 端口是否被占用使用如下命令。

```
# netstat -ntlp |grep 6379
tcp        0      0 127.0.0.1:6379          0.0.0.0:*               LISTEN
3475/redis-server
```

（2）查询 Redis 服务是否运行。

```
#ps xal|grep redis
```

（3）测试启动。返回 PONG 表示启动成功。

```
# redis-cli  ping
```

注意启动 Redis 客户端工具命令为 redis-cli。

```
Redis 127.0.0.1:6379>
```

（4）查看服务器的统计信息（以 OSSIM USM 为例）。

```
#redis-cli  info
```

在这些统计信息中，以下内容需要了解。

- uptime_in_seconds：自 Redis 服务器启动以来，经过的秒数。
- used_memory：由 Redis 分配器分配的内存总量，以字节（byte）为单位。
- used_memory_human：以可读的格式返回 Redis 分配的内存总量。
- used_memory_rss：从操作系统的角度，返回 Redis 已分配的内存总量，该值和 top 命令输出一致。
- used_memory_peak：Redis 的内存消耗峰值（以字节为单位）。
- used_memory_peak_human：以可读的格式返回 Redis 的内存消耗峰值。
- used_memory_lua：Lua 引擎所使用的内存大小（以字节为单位）。
- total_connections_received：服务器已接受的连接请求数量。
- total_commands_processed：服务器已执行的命令数量。
- instantaneous_ops_per_sec：服务器每秒钟执行的命令数量。
- keyspace_hits：命中 key 的次数。

该命令可查看配置信息及各种统计信息。

（5）查看实时转储收到的请求。

```
#redis-cli  monitor
```

```
1422506320.387463 "SETEX" "SimpleCache-system:system_all_info:7a4c186bebf9b39e951a5465f4eee3fe" "
600" "[false, \"Can't connect to system with IP 192.168.91.135 msg:{'msg': 'SSH encountered an un
known error during the connection. We recommend you re-run the command using -vvvv, which will en
able SSH debugging output to help diagnose the issue', 'failed': True} \"]"
1422506320.387608 "SADD" "SimpleCache-system-keys" "system_all_info:7a4c186bebf9b39e951a5465f4eee
3fe"
1422506320.387626 "EXEC"
1422506321.668430 "PING"
1422506324.871261 "SCARD" "SimpleCache-sensor_plugins-keys"
1422506324.871778 "MULTI"
1422506324.871800 "SETEX" "SimpleCache-sensor_plugins:get_plugins_from_yaml:95f77ee87647cb030fdee
bb77e0f53ba" "600" "[true, {\"dark\": {\"192.168.91.131\": {\"msg\": \"SSH encountered an unknown
 error during the connection. We recommend you re-run the command using -vvvv, which will enable
SSH debugging output to help diagnose the issue\", \"failed\": true}}, \"contacted\": {}}]"
1422506324.871867 "SADD" "SimpleCache-sensor_plugins-keys" "get_plugins_from_yaml:95f77ee87647cb0
30fdeebb77e0f53ba"
1422506324.871886 "EXEC"
1422506324.872778 "PING"
1422506330.513573 "SCARD" "SimpleCache-system-keys"
1422506330.593223 "MULTI"
1422506330.593278 "SETEX" "SimpleCache-system:system_all_info:4946183995572b33a09572757bd35efd" "
600" "[false, \"Can't connect to system with IP 192.168.91.131 msg:{'msg': 'SSH encountered an un
known error during the connection. We recommend you re-run the command using -vvvv, which will en
able SSH debugging output to help diagnose the issue', 'failed': True} \"]"
1422506330.593406 "SADD" "SimpleCache-system-keys" "system_all_info:4946183995572b33a09572757bd35
```

下面对操作做个简单的解释：

- MULTI　　/*事务开始*/
- EXEC　　/*执行事务*/
- PING　　/*通常用于测试与服务器的连接是否仍然生效*/
- SCARD　　/*返回集合 key 的基数（集合中元素的数量）*/
- SETEX　　/*将值 value 关联到 key，并将 key 的生存时间设为 seconds*/

（6）查看系统所有 Keys。

```
redis 127.0.0.1:6379> keys *    /*  keys 命令用于管理键 */
```

（7）查看键值类型。

type 可查看键值类型，我们看如下操作。

```
127.0.0.1:6379> keys *
 1) "SimpleCache-system:267372578e8118431d27673a097b0a291135f56c0175fd87ff367352a3f6b98a"
 2) "SimpleCache-sensor_plugins-keys"
 3) "SimpleCache-backup-keys"
 4) "SimpleCache-sensor_network-keys"
 5) "SimpleCache-network_status:141629ec54e43977e936d5683545a1d1428a0567e5b6bf0ae253404dbfb332f4"
 6) "SimpleCache-system:768395f670bf482a0791e5c86a33e1a271725bae8a728f45c396ce260e3f055d"
 7) "SimpleCache-network_status-keys"
 8) "SimpleCache-support_tunnel:b5332d8d1b59f5664757d090057d75182bc499c4602ab35647ba8a508c2f2c6a"
 9) "SimpleCache-system:a09fe08ee122adb53f5748dc49d55966bbf0159c9b1bed0f6fb1829f9771bb7e"
10) "SimpleCache-sensor_plugins:11a1ecd0ce60b5eedb8ee22dd9d83e6ad3a79a9b95a84458ae1b367391b3d68f"
11) "SimpleCache-sensor_network:0f5b6b7f461210888386719a5e3300413d37a90f8aaf3242dec02de7188f35c7"
12) "SimpleCache-backup:41380adf7f0adbf0b5516c19a6bd6fefbc06a4a4e8ea1d92e74d795b1fcfcade"
13) "SimpleCache-support_tunnel-keys"
14) "SimpleCache-ping_system-keys"
15) "SimpleCache-system_config:845b526ea96e2e234952e6c3f74275daccc33bb08451c9104338aa6b99f82efd"
16) "SimpleCache-system_config-keys"
17) "SimpleCache-system_config:97e56c5cee59d4d71186e0a9a5c2b56eabd27d324cd67d0e7c2bf08b15c2255b"
18) "SimpleCache-ping_system:9805186f8cfa4fba553145d671c650239c7cf5ec257b12d63cfaa02abc2d8a25"
19) "SimpleCache-system-keys"
127.0.0.1:6379> type SimpleCache-system-keys
set
127.0.0.1:6379> type SimpleCache-system:267372578e8118431d27673a097b0a291135f56c0175fd87ff367352a
string
```

（8）确认一个键是否存在。

```
redis 127.0.0.1:6379> exists SimpleCache-backup-keys
(integer) 1                /* 返回1表示存在 ，如果返回0 表示不存在 */
```

（9）保存最新的 key 值。

```
#redis-cli BGSAVE        /*异步保存数据到磁盘
#redis-cli SAVE          /*同步保存数据到磁盘
```

执行该命令，可在后台异步保存当前数据库的数据到磁盘。

（10）Redis 服务器性能检测。

```
ossim411:~# redis-benchmark -h localhost -p 6379 -c 100 -n 10000
====== PING_INLINE ======
  10000 requests completed in 0.30 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1
91.58% <= 3 milliseconds
97.98% <= 4 milliseconds
99.84% <= 5 milliseconds
100.00% <= 5 milliseconds
33003.30 requests per second
```

（11）安全性。

在 redis.conf 的配置文件中采用 ACL 控制器确保安全。

```
bind 127.0.0.1
```

通过上面这行语句，即可把 Redis 绑定在单个接口。有关 Redis 更多命令集使用方法，请参阅 http://redisdoc.com/。

1.7.6　Redis Server 配置详解

Memcached 是高性能分布式内存缓存服务器，本质上是一个内存 Key/Value 数据库，但不支持数据持久化，服务器关闭之后数据全部丢失，Redis 支持数据持久化。

Alienvault_api 是 OSSIM 重要组件之一，其中间件架构采用了 RabbitMQ+Celery+Redis+Erlang。Redis 也是一个开源的 Key/Value 存储系统。与 Memcached 类似，Redis 将大部分数据存储在内存中，支持的数据类型包括：字符串、哈希表、链表、集合、有序集合以及基于这些数据类型的相关操作。

Redis 使用 C 语言开发，能在大多数 Linux、BSD 和 Windows 系统应用。当前 Redis 应用广泛在新浪、淘宝、Github 等组织。

OSSIM 4.8 系统中 Redis 系统的配置文件路径为/etc/redis/redis.conf，下面我们在 OSSIM 系统中，首先来查看 Redis 的配置文件中定义了哪些主要参数以及这些参数的作用。

（1）第 17 行“daemonize yes”，默认情况下 redis 在后台运行。

（2）第 21 行“pidfile /var/run/redis/redis-server.pid”，这里 Redis 默认存放 PID 文件的位置。

（3）第 25 行“port 6379”，服务端口默认为 6379。

（4）第 30 行“bind 127.0.0.1”，只接收来自本机的请求，如果注销此行那么则处理所有

发来的请求。

（5）第 48 行“loglevel notice”，日志记录级别，Redis 共支持四个级别，分别是 debug、verbose、notice、warning，在生产环境下通常使用 notice 级别。

（6）第 53 行“logfile /var/log/redis/redis-server.log”，日志记录的位置。

（7）第 68 行“databases 16”，可用数据库。

（8）第 86 行“save 900 1”，保存数据到磁盘。Redis 支持两种持久化方式：一种是快照方式（snapshotting），是将数据存储到二进制文件中，另一种是 aof(Append only file)方式，默认未启用该方式。下面看第一种快照方式的配置，如果 Redis 在 N 秒内，超过 M 个 key 被修改，则自动做快照，其格式为 save <seconds><changes>，代表在多长时间内，达到多少次更新操作，就将数据同步到数据文件 RDB，实际上相当于条件触发取快照的机制。“save 900 1”表示 900 秒内至少有 1 个 key 被改变就保存数据到磁盘。读者可以查看/var/lib/redis/dump.rdb，该文件默认会将当前数据库中所有数据转储到文件。对于“save 900 1”这条语句表示 900 秒做一次快照。由于快照方式在一定间隔时间内完成，如果 Redis 意外宕机，就会丢失最后一次快照的内容，才出现了 Append only file 方式。

（9）第 94 行“rdbcompression yes”，存储至本地数据库时是否压缩数据，默认为 yes；

（10）第 97 行“dbfilename dump.rdb”，定义存储数据库文件名，路径在/var/lib/redis/dump.rdb。

（11）第 107 行 “dir /var/lib/redis”，定义数据库文件存放路径。

（12）第 217 行“maxmemory 536870912”，指定 Redis 最大内存设置为 512MB，也可以比这个值大，Redis 在启动时会把数据加载到内存中，达到最大内存后，Redis 会先尝试清除已到期或即将到期的 Key，Redis 同时也会移除空的 list 对象。

（13）第 240 行“maxmemory-policy volatile-lru”，当内存达到最大值的时候 Redis 会选择的数据处理方式：volatile-lru 代表利用 LRU 算法（Least Recently Used 最近最久未使用算法，一种内存回收算法）移除设置过期时间的 key。

（14）第 268 行“appendonly no”，这是另一种数据持久化方式，默认情况下，Redis 会在后台异步方式把数据库镜像备份到磁盘。如果此处设置为 yes，那么就必须启用 appendfilename appendonly.aof（此功能默认为关闭状态）。

（15）第 294 行“appendfsync everysec”，Redis 支持三种同步 AOF 文件的策略: no 代表不进行同步，由系统操作。always 代表每次只要有写操作就进行同步，everysec 代表对写操作进行累积，每秒同步一次，默认“everysec”，按照速度和安全二者进行折中。

（16）第 482 行“activerehashing yes”，Redis 将在每 100 毫秒时，使用 1 毫秒的 CPU 时间来对 Redis 的 hash（哈希）表进行重新 hash，这样可以降低内存的使用。

1.7.7 RabbitMQ、Redis 与 Memcached 监控

RabbitMQ 提供了完善的管理和监控工具，下面在 OSSIM USM 中开始设置 Web 监控。

1. RabbitMQ Web 监控

```
#rabbitmq-plugins enable rabbitmq_management        /*开启插件*/
#/etc/init.d/rabbitmq-server restart          /*重启 rabbitmq-server 服务*/
#netstat -nlp |grep beam               /* 查看端口是否被监听* /
#tail -f /var/log/rabbitmq/alienvault\@localhost-sas1.log   /*查看日志*/
#rabbitmqctl add_user ossim a1b2c3     /*设置管理用户（ossim），密码（a1b2c3）*/
#rabbitmqctl set_user_tags ossim monitoring  /*设置管理员的角色*/
#rabbitmqctl list_users                /*查看设置用户是否正确*/
```

在浏览器输入 http://yourip:15672/进行访问，如果从远程登录用户名和密码可以采用以上指定的 ossim/a1b2c3，如果是本机登录可以采用 guest/guest。通过 Web 界面实时掌握 RabbitMQ 服务器文件、队列、进程、内存和磁盘空间占用情况。

2. Redis 监控

通常通过 Redis 自带的 info、monitor 命令获取信息，操作不方便。如何对它进行监控，了解它的性能，我们先通过一个命令行工具 redis-faina 来感受一下效果。

```
#mkdir /test
#cd /test
#apt-get install git
#apt-get install python-pip
#pip install argparse
#git clone https://github.com/Instagram/redis-faina.git
#cd redis-faina/
#redis-cli -p 6379 MONITOR | head -n 100 | ./redis-faina.py --redis-version=2.4
```

该命令执行后可看到实时的数据，并且有一定的统计数据功能，可以作为命令行工具使用，不过 Redis 版本要大于 2.4。

3. Redis-live

Redis Live 是另一款用来监控 Redis 实例，分析查询语句，比 redis-faina 更方便的是它具有 Web 界面，更加直观。要运行 redis-live 首先安装它所需要的所有依赖包，步骤如下：

```
#easy_install pip                /*安装 Python 一个公共资源库*/
#pip install tornado             /*安装 tornado，它是一款轻量级 Web 服务器*/
#pip install redis               /* 安装 Python 的 redis 模块*/
#pip install python_dateutil     /*安装 Python 第三方模块*/
#pip install argparse
```

下面从远程主机克隆版本库。

```
#git clone https://github.com/nkrode/RedisLive.git
#cd RedisLive/src
#cp redis-live.conf.sample redis-live.conf    /*备份配置文件*/
```

接着编辑 redis-live.conf，默认只能本机访问，所以配置文件中的 IP 为 127.0.0.1。

```
#vi redis-live.conf
"RedisServers":
{
"server": "127.0.0.1 ",
"port": 6379
} 被监控的 Redis，可以配置多个。
"DataStoreType" : "redis", ----监控数据存储方式

"Redisstats server":
  "server": "192.168.11.100",
  "port": 6380    存储监控数据的 Redis
```

启动 redislive。

启动监控脚本，监控 30 秒，duration 参数以秒为单位。启动命令如下：

```
#./redis-monitor.py  --duration=30
```

启动 WebServer。

Redislive 使用 tornado 作为 Web 服务器。启动命令如下：

```
#./redis-live.py
```

浏览器输入 http://ip:8888/index.html，这里 ip 换成你的 IP 地址就 OK 啦。

4. phpMemcachedAdmin

phpMemcachedAdmin 是一个用来管理 memcached 的一个可视化的管理工具，采用 php 的脚本语言实现，下载地址为 http://down.51cto.com/data/2098348，操作步骤如下。

安装很简单，直接解压缩输入服务器地址端口和路径即可使用，下面新建一个目录。

```
#mkdir /usr/share/ossim/www/phpmemcached/
#tar zxvf phpMemcachedAdmin-1.2.1-r233.tar.gz -C /usr/share/ossim/www/phpmemcached/
```

然后在浏览器地址栏中输入 https://yourip:11211/ossim/phpmemcached/，即可访问它的 Web UI 界面。

1.8 OSSIM 高可用架构

随着 OSSIM 在企业中深入部署，我们会考滤如何提高 OSSIM 系统的高可用性。一台 OSSIM 存在单点故障暗藏安全隐患，可以采用 MySQL_HA（MySQL 双机热备）的手段得以实现。

1.8.1 OSSIM 高可用实现技术

Linux 下常通过 Keepadlived+LVS 或者 Heartbeat+LVS 的方案实现 HA，Heartbeat 项目是 Linux-HA 工程的组成部分，是目前开源 HA 项目中最成功的例子，而且 Heatbeat 也是 OSSIM

系统自带的组件（默认已安装完成 V3.0.3）。如图 1-47 所示为 OSSIM HA 的架构图。

图 1-47　OSSIM-HA 架构图

这里安装 HA 有两个步骤，一个为实体应用，另一个为备用的虚拟应用。至少要保持两台 Server 的 AlienVault-Center 之间连接，因为 AlienVault-Center 管理着 Agent、Server、IDM、Forwarder 等服务，所以需要它们和虚拟 IP 保持连接。

保留从服务器（OSSIM）的以下服务：

- Alienvault Center
- Rsyslog
- Cron
- Heartbeat
- MySQL
- MongoDB

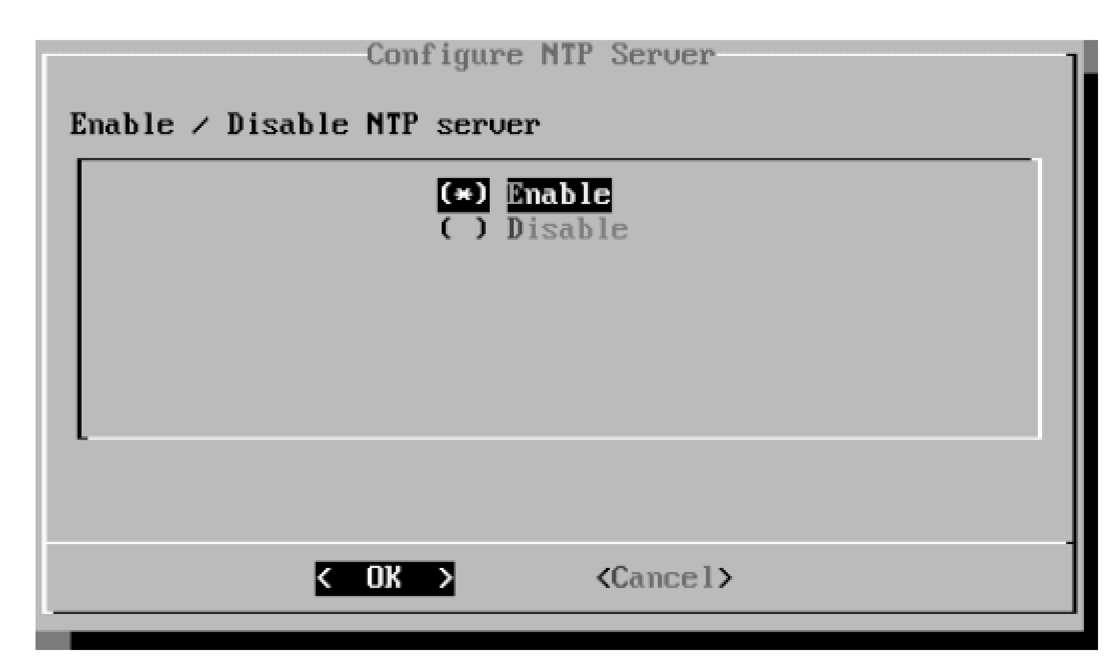
OSSIM-HA 地址分配举例如表 1-9 所示。

表 1-9　OSSIM-HA 地址分配举例

<table>
<tr><th>主机名</th><th>真实 IP</th><th>虚拟主机名</th><th>虚拟 IP</th><th>报告</th></tr>
<tr><td>HA-Server1</td><td>192.168.207.110</td><td rowspan="2">VServer</td><td rowspan="2">192.168.207.120</td><td rowspan="2">192.168.207.121</td></tr>
<tr><td>HA-Server2</td><td>192.168.207.111</td></tr>
<tr><td>HA-Logger1</td><td>192.168.207.112</td><td rowspan="2">VLogger</td><td rowspan="2">192.168.207.121</td><td rowspan="2">None</td></tr>
<tr><td>HA-Logger2</td><td>192.18.207.113</td></tr>
<tr><td>HA-Sensor1</td><td>192.168.207.114</td><td rowspan="2">VSensor</td><td rowspan="2">192.168.207.121</td><td rowspan="2">192.168.207.120</td></tr>
<tr><td>HA-Sensor2</td><td>192.168.207.115</td></tr>
</table>

1.8.2　安装环境

Master/Slave 所有 OSSIM（v4.3）系统版本保持一致，即要求两台服务器软件硬件配置一致，两台服务器同在一个网段。安装期间分配每台设备一个 IP，当系统安装好后，首先进行同步时间进入控制台 System Preferences→Change Location → Change Location→Date and time → Configure NTP Server，修改方法如图 1-48 所示。



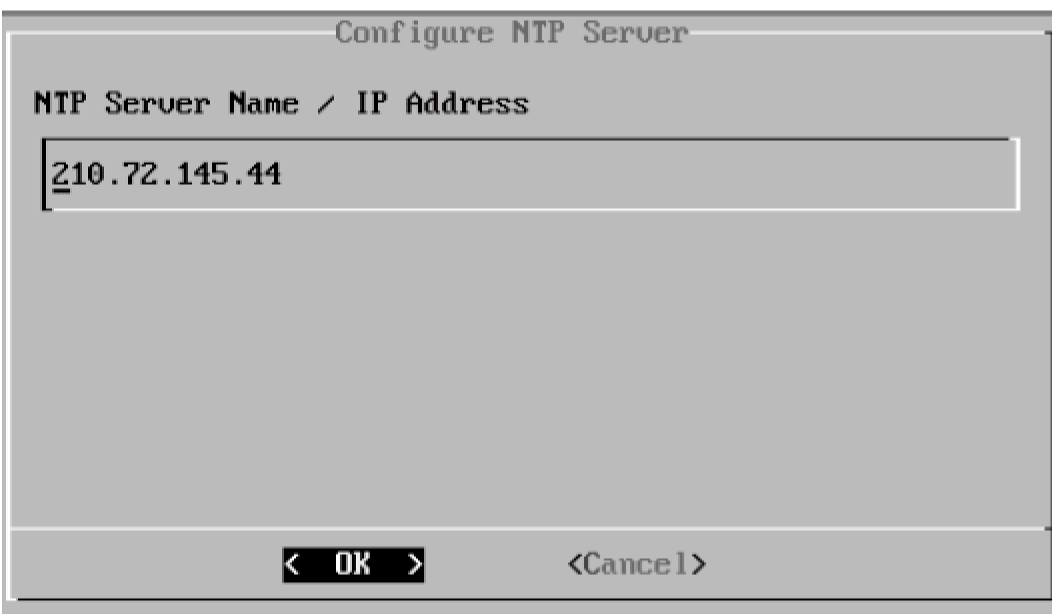


图 1-48　配置 NTP

然后修改默认分配的主机名，注意不要重名，最后选择 Apply all changes 确认。

1.8.3　配置本地主机

首先 SSH 远程登录或者直接在服务器的控制台下操作，在终端控制台下选择“Jailbreak System”，编辑 ossim_setup.conf 配置文件。

```
#vi /etc/ossim/ossim_setup.conf
```

修改以下域值：

```
Ha_heartbeat_start=no 改为“yes”。
Ha_local_node_ip= 输入本地应用服务器 IP。
Ha_other_node_ip= 输入远程应用服务器 IP。
Ha_other_node_name= 输入远程应用服务器主机名称。
Ha_password= 输入口令，这个口令对于本地和远程应用服务器主机是相同的口令。
Ha_role= 输入‘master’。
Ha_virtual_ip=输入分配给虚拟设备 IP 地址（默认是 unconfigured，未配置）。
```

保存退出后，进行以下操作。

```
#ossim-reconfig
```

1.8.4　配置远程主机

同样方法，编辑远程主机的 ossim_setup.conf 文件。

```
#vi /etc/ossim/ossim_setup.conf
```

修改以下域值。

```
Ha_heartbeat_start= 改为 “yes”。
Ha_local_node_ip= 输入远程 IP。
Ha_other_node_ip= 输入本地 IP。
Ha_other_node_name= 输入本地应用名称。
Ha_password=输入口令，注意这个口令对于本地和远程应用服务器主机是相同的口令
Ha_role= 输入“slave”。
Ha_virtual_ip=输入分配给虚拟设备 IP 地址。
```

保存退出后执行如下命令。

```
#ossim-reconfig
```

1.8.5　同步数据库

连接本地应用服务器，进入 root 控制台，输入如下命令：

```
#ossim-reconfig  --mysql_replication
```

1.8.6　同步本地文件

在同步本地文件时，需注意下面的操作必须在本地和远程服务器上完成。开始编辑 /etc/cron.d/ossim_ha_rsync，为了保证两台机器之间的同步，需要激活下面配置行。

```
Risk metrics graphs
#0 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_lib_ossim_rrd /var/lib/ossim/rrd >/dev/null

Netflow configuration
#2 * * * * root /usr/local/sbin/ossim_ha-rsync.sh etc_nfsen /etc/nfsen >/dev/null

Netflow data collected
#4 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_cache_nfdump /var/cache/nfdump >/dev/null

Netflow graphs
#6 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_nfsen /var/nfsen >/dev/null

Nagios configuration
#8 * * * * root /usr/local/sbin/ossim_ha-rsync.sh etc_nagios3_conf.d/etc/nagios3/conf.d >/dev/null

Nagios checks
#10 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_cache_nagios3 /var/cache/nagios3 >/dev/null

Ossim database backups
#12 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_lib_ossim_backup /var/lib/ossim/backup >/dev/null

Ossim agent configuration and plugins
#14 * * * * root /usr/local/sbin/ossim_ha-rsync.sh etc_ossim_agent/etc/ossim/agent >/dev/null

Ntop graphs and statistics
#16 * * * * root /usr/local/sbin/ossim_ha-rsync.sh var_lib_ntop /var/lib/ntop >/dev/null

Ntop configuration
#18 * * * * root /usr/local/sbin/ossim_ha-rsync.sh etc_ntop /etc/ntop >/dev/null
```

经过以上几个步骤 HA 配置工作即告一段落。

1.9 OSSIM 防火墙

Linux 中很多功能都是以模块加载方式来扩充系统功能，Netfilter 同样采用这种方式存在于 Linux 中。如果理解了 Linux 模块加载也就能够理解 Netfilter 的模块加载方式。大家在/lib/modules/kernel_version/kernel/net/ipv4/netfilter/目录下能够看到很多以“.ko”为扩展名的文件，那里存放的就是已经编译好的模块文件。如果在 Netfilter 模块目录仔细查看，会发现各种功能的模块，这些模块仅是提供某些过滤功能，如果想让 Netfilter 为我所用，还需要赋予它执行的“规则”，定出了规则后，Netfilter 才知道哪些数据包通过，哪些数据包阻止，又有哪些数据包进行置换处理。

1.9.1 理解 Filter 机制

首先我们看个例子，在客户机上通过 Firefox 访问，远端 Apache 服务器上的 Web 页面，像这样的操作中对于 Netfilter 有三种封包类型（如图 1-49 所示）分别是：

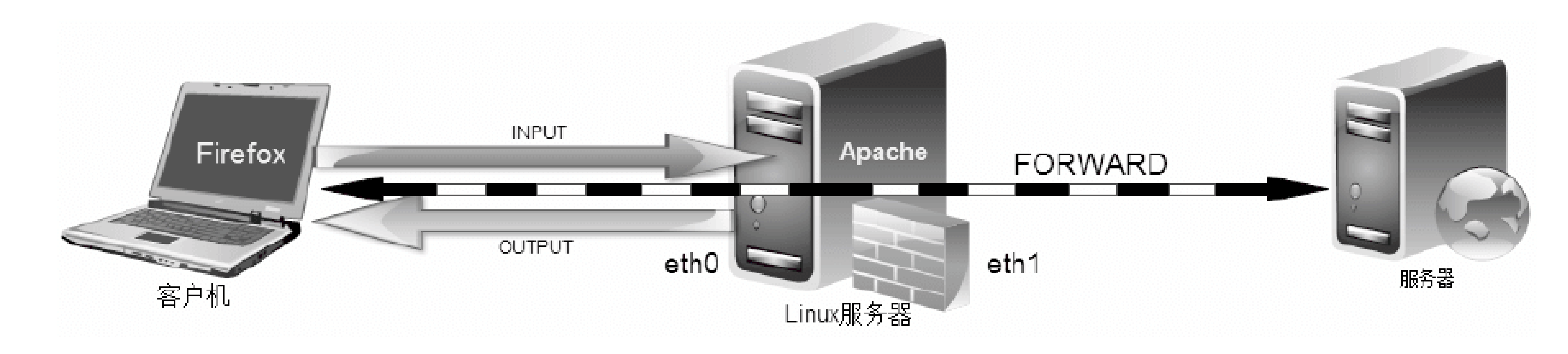


图 1-49　封包类型

1. INPUT 类型

当客户机访问 Web 服务器的 httpd 进程时，对于 Web 服务器而言，属于入站的包，换句话说，就是网上其他主机送给本机 httpd 进程的封包，这里把它定义为 INPUT 类型封包（图 1-49 中左侧上方的箭头）。

2. OUTPUT 类型

与 INPUT 相反，由 Web 服务器的 httpd 进程连接到客户机上的这类封包就属于 OUTPUT 类型（图 1-49 中右侧上方的箭头），也就是本机 httpd 进程所产生的封包。

3. FORWARD 类型

在图 1-49 中，如果是 Web 服务器的封包，就属于 FORWARD 类型。在什么情况下会产生这种类型封包？我们拿 Linux 当作路由器使用时就会有 FORWARD 类型封包产生。

有了上面的基本概念，下面开始讨论过滤表，Filter Table 有三个链（Chain），大家可以形象地理解为自行车中的链条，它是环环相扣的，封包在网络上传送也是如此。

Filter 表的结构如图 1-50 所示，你会发现三种封包类型，下面详细为大家分析一下：

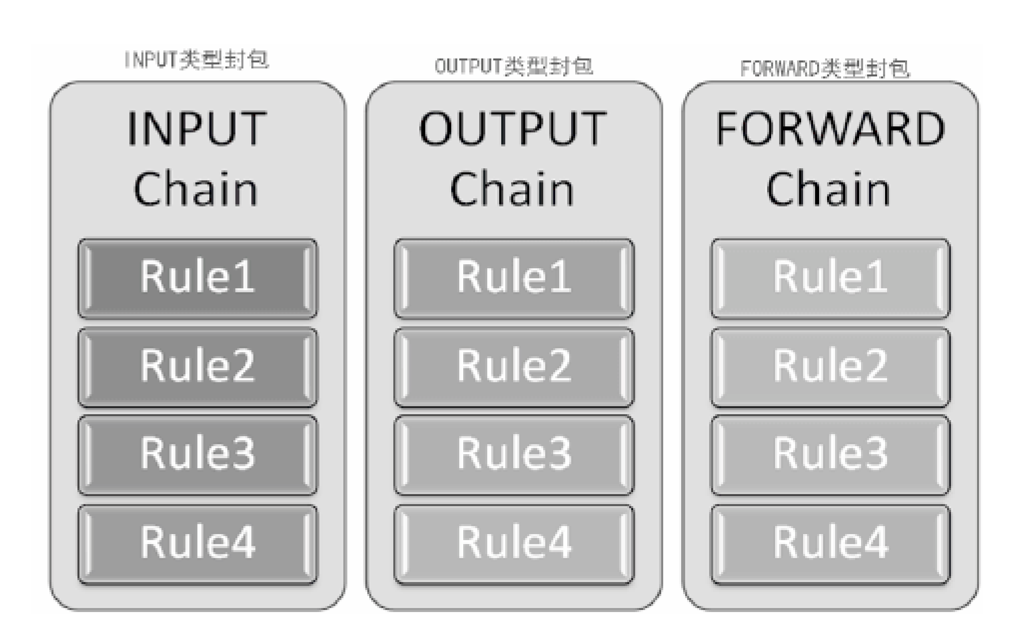


图 1-50　Filter 表结构

- INPUT 链：为了理解这一问题，我们还是拿个实例进行分析，例如我们要保护 Web 服务器的 httpd 进程，在三种封包类型中，应该注意哪一种呢？肯定是 INPUT 类型封包，那么你需要将用来过滤 INPUT 类型的封包写到 INPUT 链的规则之中，这样就能起到保护 httpd 进程的目的。总之，INPUT 链是用来存放过滤 INPUT 类型封包的规则，常用在保护本机的情况下。
- OUTPUT 链：OUTPUT 链是用来存放过滤 INPUT 类型封包的规则，常用于限制本机进程的网络链接。
- FORWARD 链：在图 1-50 中 Linux 服务器作为路由器使用，用来保护远端的一台 Web 服务器，那么应该限制那类封包类型？就是 FORWARD 类型，也就是说 FORWARD 链用来保护防火墙后面的服务器。

由于 Filter 机制非常复杂，为了使大家容易理解，用了一个简单的示意图表示 Filter 原理，如图 1-51 所示，注意图中路由表虽然画了两个，但实际只有一个，也就是我们在系统中用“route –n”看到的内容，路由表决定了封包的传送路径，它在多网卡的系统中尤为重要。本地进程就是上面所述的 Web 服务器进程，大家可以理解为 httpd 进程。

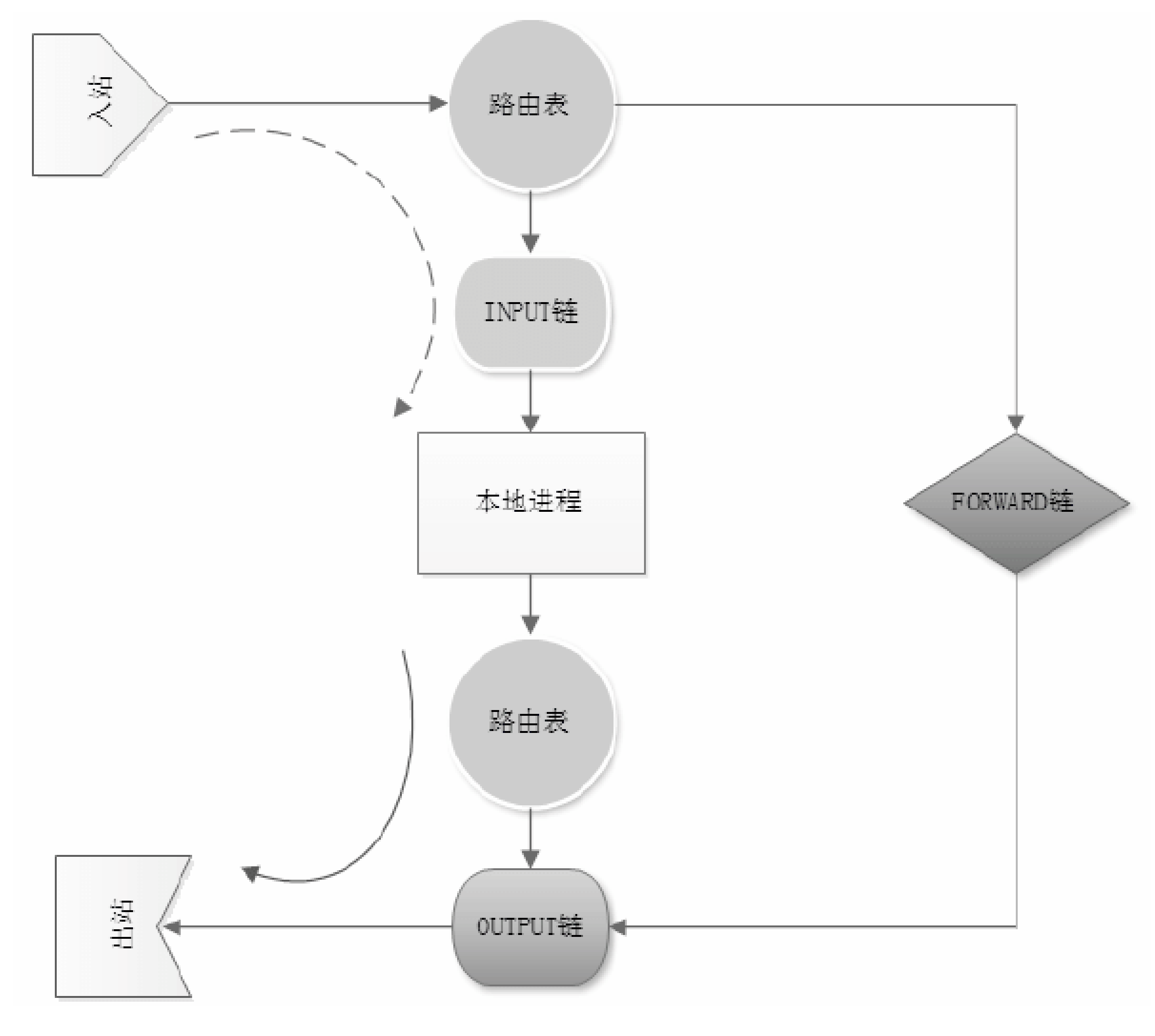


图 1-51　Filter 原理图

接下来，以三种不同的场景来说明网络封包在 Filter 机制中被处理的过程。

（1）在上网时，客户机连接服务器，也就是网络封包的目的地是 Web 服务器的 httpd 进程，封包首先送到路由表，并由路由表的内容决定传输路径。既然客户机是访问 Web 服务器上的页面，也就是封包是送给本地进程 httpd 的，因此封包被送入 INPUT 链，此时如果 INPUT 链允许通过，那么封包就会被送入本地进程，不让过就被丢弃（Drop 掉）。

（2）当服务器返回结果，需要链接客户机进程，这时由本地进程 httpd 往外传送封包，首先封包会被传送入路由表，由这里的内容决定封包传输路径，接着被送入 OUTPUT 链，如果允许则出站，如果不允许则封包就会被丢弃（Drop 掉）。

（3）当 Linux 系统作为防火墙部署，当作网关式防火墙使用，当封包要通过防火墙，首先入站进入路由表，由路由表判断，封包要由另一网卡口送出，封包就会送入 FORWARD 链，此时如果 FORWARD 链里的规则不许封包通过，那么直接丢弃。反之，封包出站离开防火墙。

1.9.2 规则匹配过程

介绍了 Filter 工作原理之后，接下来讲解封包在每个链中如何匹配。这个概念也非常重要。还是以图 1-51 为例加以说明，实际应用中每个链中包含的规则数量不尽相同，无论哪一个 Filter 表其匹配原则都是“First Match”，即优先执行。当我们在防火墙上添加的新规则被逐条加入到 INPUT 链中，被顺序编号，例如 rules1、rules2 等。当封包进入 INPUT 链之后，Filter 机制会根据这个封包的特征从 INPUT 链内的第一条规则逐个往下匹配，如果封包进来遇到第一条规则允许通过，那么这个封包就会进入到本地进程 httpd，而不管 rule2、rule3 的规则是什么。相反，如果第一条规则要丢弃，即便是 rule2 规则允许通过，也不起任何作用，这就是“First Match”原则。

在使用规则时大家要注意一个原则，尽量减少不必要的规则，因为当封包进入防火墙后会在特定的链里逐一被对比，规则条数越多，封包在防火墙中滞留的时间就越长，消耗 CPU 运算周期越长，防火墙性能就会降低。

防火墙规则顺序也会影响其工作效率，例如客户端通过 Linux 防火墙收取外网邮件， 客户端使用程序为 Outlook，使用的是 Pop3 协议。防火墙规则如下：

```
(1) iptables -A INPUT -p tcp --syn -m state --state NEW --dport 22 -j ACCEPT
(2) iptables -A INPUT -p tcp --sync -m state --state NEW --dport 25 -j ACCEPT
(3) iptables -A INPUT -p tcp --sync -m state --state NEW --dport 80 -j ACCEPT
(4) iptables -A INPUT -p tcp --sync -m state --state NEW --dport 110 -j ACCEPT
(5) iptables -A INPUT -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
```

当客户端的第一个封包需要花费 4 次匹配动作，如果一封邮件需要 1000 个封包的话，再加上 TCP 三次握手的封包这样的匹配的次数达到数万次之多，如果将第四条规则移到第一条，那么匹配次数只需 1000 多次，由此可见规则的顺序对于防火墙性能有着很大影响。所以需要将使用频度最高的规则写在第一个，以后以此类推。那么，如何确定哪个规则使用频率高

或者低呢？可使用下面命令查看：

```
alienvault:~# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0            icmp type 13
 5239  499K ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
1189K  403M ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
67007   88M ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
   11   620 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:40007
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:33800
```

还有特殊情况，封包从 INPUT 链进来，从第一条规则匹配到最后一条都没有匹配成功，怎么办？系统已有考虑，在每个链的最后有一条默认策略，而且这条默认策略总是最后才被匹配，而且这条默认策略的状态只有一种（ACCEPT 或 DROP），我们假设默认策略为 ACCEPT，那么封包这时可以进入到本地进程 如果为 DROP，则被默认策略丢弃。最后强调一点，Netfilter 的默认策略预设值为 ACCEPT，这也可以修改。我们查看 40001 端口是否在监听状态，可以使用 netstat 命令，也可以使用 iptables，应用如下：

```
alienvault:~# netstat -tanep |grep 40001 |grep LISTEN
tcp        0      0 0.0.0.0:40001           0.0.0.0:*               LISTEN      107        86130       20412/ossim-serve
r
alienvault:~# iptables -L -xvn |grep 40001
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:40001
alienvault:~#
```

1.9.3　iptables 规则库管理

我们知道 iptables 的规则被存储在各种不同的链中，相当于一个规则库。当我们修改完规则可用 iptables-save 命令进行存储。

```
#iptables-save  > /etc/ossim_firewall   /*将防火墙规则保存输出到文件*/
```

这样操作后，对于 OSSIM 系统而言，iptables 规则存储在/etc/ossim_firewall 文件中，每次启动系统时，会通过 iptables-restore 调用/etc/ossim_firewall 文件到内存中，但是这样管理未必是好事，比如，规则库中有 100 条规则，其中有 80 条规则都包含地址 192.168.11.10，那么如果需要修改将原来的 192.168.11.10 改成 192.168.12.10，你打算将所有的地址都重新输入一遍吗？这样效率太低。可建议大家采用脚本管理方式，也就是将规则写入 Shell 脚本，这样可以将 IP 地址存储变量中。重新加载防火墙命令如下：

```
#alienvault-firewall-reload
#iptables -S                   /*注意是大写字母 S，这相当于--list-rules*/
```

如果读者掌握以上内容后，对于今后需要在 OSSIM Server 上开放某个端口应该不会有问题，另外，系统在/etc/ossim/firewall_include 文件中给出了模板，参照此模板编写 iptables 规则会简单许多，而在/etc/iptables/目录下的规则，可以将各种服务的通信端口和防火墙规则对应起来。

1.10 OSSIM 的计划任务

计划任务是系统在约定的时间，执行已经计划好的工作。在 Linux 中我们经常用到 cron 来完成这项工作。cron 服务器可以根据配置文件约定的时间来执行特定的任务。当 cron 启动后，会读取所有配置文件（全局性配置文件/etc/crontab，以及每个用户的计划任务配置文件中），然后根据命令和执行时间来按时调度任务。

1.10.1 Linux 计划任务

crontab 用于设置周期性被执行的任务（指令）。该命令从标准输入设备读取指令，并将其存放于“crontab”文件中，图 1-52 展示了时间表如何分配。

图 1-52 crontab 各区域分配含义

- A1: minute — 分钟，从 0 ~ 59 的任何整数。
- A2: hour — 小时，从 0 ~ 23 的任何整数。
- A3: day — 日期，从 1 ~ 31 的任何整数。
- A4: month —月份，从 1 ~ 12 的任何整数（或使用月份的英文简写如 Jan、Feb 等）。
- A5: week — 星期，从 0 ~ 7 的任何整数，这里的 0 或 7 代表星期日（或使用星期的英文简写如 Sun、Mon 等）。
- A6: 脚本 — 要执行的命令（命令可以是 ls /proc >> /tmp/proc 之类的命令）。

在 A1~A5 的框中，可填星号（*），用来代表所有有效的值。比如，月份值中的星号意味着在满足其他制约条件后每月都执行该命令。 整数间的短线（-）指定一个整数范围。比如，1-5 意味着整数 1、2、3、4、5 用逗号（，）隔开的一系列值指定一个列表。比如，1、2、3、4、5 表明这 5 个指定的整数。

正斜线（/）可以用来指定间隔频率。在范围后加上 /<integer>意味着在范围内可以跳过 integer。比如，0-59/2 可以用来在分钟字段定义每两分钟。间隔频率值还可以和星号一起使用。比如，*/3 的值可以用在月份字段中表示每 3 个月运行 1 次任务。

- /etc/cron.hourly: 代表每小时执行脚本;
- /etc/cron.daily: 代表每天执行脚本;
- /etc/cron.weekly: 代表每周执行脚本;
- /etc/cron.monthly: 代表每月执行脚本。

如果某个任务需要根据调度来执行，而不是每小时、每天、每月执行，那么可以添加到

/etc/cron.d 目录下，该目录中所有文件和/etc/crontab 中应用相同语法结构。下面以 OSSIM 系统举例说明：

例 1：

```
alienvault:/etc/cron.d# cat alienvault_ip_reputation
15 * * * * root /usr/share/ossim-installer/update_reputation.py
```

代表：每月每天每小时的第 15 分钟执行一次 update_reputation.py 脚本。如果当前时间为 2014-12-28 22:15:00，那么接下来 2 次执行时间是：

- 2014-12-28 22:15:00
- 2014-12-28 23:15:00
- 2014-12-29 00:15:00

例 2：

```
alienvault:/etc/cron.d# cat ossim-scanner-job
*/1 * * * * root  /usr/share/ossim/scripts/vulnmeter/nessus_jobs.pl -c >>
/var/log/ossim/nessus_jobs 2>&1
```

每隔 1 分钟执行 nessus_jobs.pl 脚本，但不显示在屏幕上。那么执行该脚本后，下 3 次执行时间是：

- 2014-12-28 21:44:00
- 2014-12-28 21:45:00
- 2014-12-28 21:46:00

例 3：

```
alienvault:/etc/cron.d# cat php5
09,39 *     * * *     root   [ -x /usr/lib/php5/maxlifetime ] && [ -d /var/lib/php5 ]
&& find /var/lib/php5/ -type f -cmin +$(/usr/lib/php5/maxlifetime) -delete
```

第 9 分钟和第 39 分钟执行一次后面的脚本。那么执行该脚本后，3 次时间是：

- 2014-12-28 22:09:00
- 2014-12-28 22:39:00
- 2014-12-28 23:09:00

例 4：

```
alienvault:/etc/cron.d# cat sysstat
5-55/10 * * * * root command -v debian-sa1 > /dev/null && debian-sa1 1 1
```

表示从第 5 分钟到第 55 分钟，每 10 分钟执行一次，共可以执行 4 次。那么执行该脚本后，下 4 次时间是：

- 2014-12-28 21:55:00

- 2014-12-28 22:05:00
- 2014-12-28 22:15:00
- 2014-12-28 22:25:00

例 5：

```
alienvault:/etc/cron.d# cat av_update_cpe
0 23 * * *  root /usr/bin/alienvault-update-cpe >/dev/null
```

每晚 23 点执行 alienvault-update-cpe 脚本。那么执行该脚本后，下 3 次时间是：

- 2014-12-28 23:00:00
- 2014-12-29 23:00:00
- 2014-12-30 23:00:00

1.10.2 OSSIM 中的计划任务

1. Cron.d 全局计划任务

当我们要增加全局计划任务时，可直接修改/etc/crontab。/etc/cron.d 目录就是为了解决这种问题而创建。以下计划任务采用 OSSIM 4.8 平台来讲解。/etc/crontab 为系统 crontab 配置文件，通常只能被 root 用户和守护进程用来设置系统类的 cron 任务，所有 Linux 用户须使用 crontab 来创建、编辑 cron 任务。

例如，增加一项定时备份任务，可以这样处理，在/etc/cron.d 目录下新建文件 Update_sensors 内容如下：

```
   0 * * * *     root [ -x /usr/bin/alienvault-update_sensors ] &&
/usr/bin/alienvault-update_sensors
```

该目录下还有如表 1-10 所示的重要的计划任务。

表 1-10 OSSIM 全局计划任务

脚本名称： alienvault_ip_reputation 15 * * * * root /usr/share/ossim-installer/update_reputation.py
alienvault_ip_reputation_feedback 30 * * * * root /usr/share/ossim/scripts/send_reputation_feedback.py
脚本名称：av_update_cpe 0 23 * * *   root /usr/bin/alienvault-update-cpe >/dev/null
脚本名称：munin */5 * * * *   munin if [-x /usr/bin/munin-cron]; then /usr/bin/munin-cron; fi 14 10 * * *   munin if [-x /usr/share/munin/munin-limits]; then /usr/share/munin/munin-limits --force --contact nagios --contact old-nagios; fi
脚本名称：Munin-node */5 * * * * root if [-x /etc/munin/plugins/apt_all]; then /etc/munin/plugins/apt_all update 7200 12 >/dev/null; elif [-x /etc/munin/plugins/apt]; then /etc/munin/plugins/apt update 7200 12 >/dev/null; fi

脚本名称：OSSIM-cd-tools 0 1 * * * root /usr/bin/unauto-apt > /dev/null 0 2 * * * root /usr/bin/apt-get autoclean > /dev/null
脚本名称：OSSIM-scanner-job */2 * * * * root /usr/share/ossim/scripts/vulnmeter/nessus_jobs.pl -c >> /var/log/ossim/nessus_jobs 2>&1
脚本名称：Php5 09,39 * * * * root [-x /usr/lib/php5/maxlifetime] && [-d /var/lib/php5] && find /var/lib/php5/ -type f -cmin +$（/usr/lib/php5/maxlifetime）-delete
脚本名称：Update_sensors（每小时运行一次） 0 * * * * root [-x /usr/bin/alienvault-update_sensors] && /usr/bin/alienvault-update_sensors

cron 进程执行时，会自动扫描该目录下的所有文件，按照文件中的时间设定执行后面的命令。cron 执行时，也就是要读取三个地方的配置文件：

- /etc/crontab。
- /etc/cron.d 目录下的所有文件。
- 每个用户的配置文件。

需要注意的是，这些计划任务脚本文件的所有者和权限都是 root，该任务都是以 root 身份完成。

2. 添加调度任务

管理员可在/etc/cron.daily/、/etc/cron.hourly/、/etc/cron.weekly/中添加调度，这几个目录系统已经设置好，定时运行着每天、每周、每月、每小时的调度任务。所以可以简单地将需要执行的调度任务，存放到相应的目录中，cron.hourly 每小时运行一次该目录中所有脚本，如表 1-11 所示。

表 1-11　每小时计划任务

脚本名称：ossim-compliance /usr/bin/perl -I"/usr/share/ossim/compliance/scripts/datawarehouse/perl" "/usr/share/ossim/compliance/scripts/datawarehouse/OSSIM_ETL.job_ReportingETL.pl" --context=Default
脚本名称：ossim-compliance-iso27001 /usr/bin/perl /usr/share/ossim/compliance/scripts/datawarehouse/iso27001sid.pl
脚本名称：ossim-sem sh /usr/share/ossim/scripts/sem/forensic_hourly.sh
脚本名称：scheduler-report /usr/bin/php /usr/share/ossim/scripts/report/launcher.php >> /tmp/logscheduler 2>&1

cron.daily 每天要执行的脚本文件，如表 1-12 所示。

表 1-12　每日计划任务

脚本名称：alienvault-passvulnupdate （升级 CVE） cd /usr/share/ossim-cd-tools/cve/ wget -N http://data.alienvault.com/cve/nvdcve-2.0-current.xml.bz2 bunzip2 nvdcve-2.0-current.xml.bz2
脚本名称：check-tickets /usr/bin/php /usr/share/ossim/scripts/incidents/check_tickets.php
脚本名称：clean-alarms # DELETE ALARMS /usr/bin/php /usr/share/ossim/scripts/alarms/delete_alarms.php # ASSET SYNC DUMP /usr/share/ossim/scripts/assets_sync.sh
脚本名称：logrotate test -x /usr/sbin/logrotate || exit 0 /usr/sbin/logrotate /etc/logrotate.conf

cron.weekly 每周运行一次该目录下的所有脚本，如表 1-13 所示。

表 1-13　每周计划任务

脚本名称：alienvault-ossec find /var/ossec/logs/alerts/ -mtime +7 -type f -exec rm {} \;　删除/var/ossec/logs/alerts 目录下七天前的文件 find /var/ossec/logs/alerts/ -type d -empty -exec rmdir {} \;　寻找/var/ossec/logs/alerts 目录下将空文件夹删除 chown www-data:ossec /var/ossec/logs/ -R chmod ug+w /var/ossec/logs -R

1.11 小结

本章从 SIM 一直介绍到 OSSIM 系统，阐明了各系统的差异，文中重点讲述了 OSSIM 发展及系统架构和工作原理，尤其是各种安全插件的使用，代理的安装和使用以及关联引擎的作用，系统数据库和端口的分布等重要内容。同时也讲解了 RabbitMQ 中间件系统在高性能 OSSIM 系统中的作用，OSSIM 高可用架构以及 OSSIM 系统的计划任务列表的工作情况。理解本章介绍的内容对于下一章安装部署 OSSIM 系统有着重要意义。

第 2 章

◂ OSSIM部署与安装 ▸

从本章节可以学习到：

- OSSIM 安装策略
- 分布式 OSSIM 体系
- 安装前的准备工作
- OSSIM 服务器选型
- OSSIM USM 与传感器的区别
- OSSIM Web 前端初始化方法
- VPN 连接
- 分布式部署实战
- 系统安装后续工作
- OSSIM 远程管理工具
- 安装桌面环境
- Alarm 报警分析
- 详解 SIEM 控制台
- 常见 OSSIM 安装错误分析

2.1 OSSIM 安装策略

正确部署 OSSIM 系统非常重要，通常我们要考虑几个问题：

- 应该监控什么？有些使用者想通过 Snort 监控所有的计算机和网络设备，这是不现实的，一般我们只用它来监控重要的服务器及网络设备。
- 哪些报警可以忽略？
- 通常，Snort 架设的 IDS 系统会产生非常多的报警，哪些需要我们关注呢？
- 传感器放置在何处？
- 如果发现违规行为如何处理？

2.1.1 未授权行为

在 OSSIM 系统中，以 Snort（最新版本为 Suricata）为主的 IDS 是一个重要功能，主要提供针对网络的异常行为和可疑行为监控，在 OSSIM 系统中通过配置 Snort 的特征码对未授权的行为进行监控并报警。什么是未授权行为？通常对于一个基于网络的入侵检测系统而言，不被允许的行为分为：恶意行为、可疑行为、异常行为和不适当行为，它们都属于未授权行为。

（1）恶意行为（Malicious behavior）

一般来讲，所有的未授权行为就是恶意行为。比如远程特洛伊木马流量，用户提权，DOS 服务攻击。

（2）可疑行为（Suspicious behavior）

我们知道 SYN 表示建立连接时使用的握手信号，而 FIN 则表示终止链接，从逻辑上讲一个包不应该同时包含 SYN 和 FIN，如果发现这种包的流量，那么很有可能是攻击者蓄意构造的（利用 nmap 就能制定使用 IP 分片方式发送碎片探测包），它具有可疑行为。单纯的就 SYN 和 FIN 包本身而言并没有恶意行为，但这两种包放到一起进行试探，即构成了可疑行为。由此推广开来，只要有人试图在很短时间内访问了大量主机和端口，表明是可疑行为。

（3）异常行为（Anomaly Behavior ）

与正常行为不同的称为异常行为，注意异常不等同于安全事故，但有可能是事故发生的早期征兆。我们把事故反过来念就是故事，每一起网络安全的事故背后都有一段故事，作为网络安全人员就需要了解其中的玄机，不能麻痹大意。

我们通过实例去理解它，内网里有一台没有安装 TFTP 服务的服务器，但发现了 TFTP 流量，这就是异常行为。在没有开放文件共享的 Windows Server 中出现了向外网的文件共享访问，监测到 139、135 端口 TCP 连接的情况这也属于异常行为。

以上两种行为就不属于入侵的前兆，而是入侵行为已经发生的标志。可能攻击者已经成功地攻击了一台主机，但没有触发报警（当然也不排除是内部用户利用合法权限攻击了主机）。如果企业采用分布式方式部署了 OSSIM 或类似工具即能捕获这些异常行为。

（4）不适当行为（Inappropriate Activity）

有些未授权行为，既不是恶意行为也不是有害的，但对组织而言属于未授权行为，这类行为可能带来麻烦。P2P 的安全问题由来已久，P2P 软件和技术本身可能是无害的，但共享的资源文件中却可能因为存在漏洞而被利用，由于其无中心化以及自分发节点不可控的特性，成为企业资料泄漏的重要途径。网络攻击者能利用 P2P 的技术弱点发动网络攻击，或者成为恶意软件，病毒传播，不法信息传播的温床。例如 P2P 文件下载或共享，这种行为能穿透防火墙达到传输秘密文件的目的，在一些 Malware 事件中通过这类程序会专门繁殖蠕虫和病毒（例如，P2P 蠕虫 Worm.Win32.Palevo.arxz，就是通过 P2P 共享文件传播自身）。该蠕虫还会自动连接 http://188.***.27/jebacina/418.exe，下载最新的版本，实现自动更新。

2.1.2 传感器位置

OSSIM 的传感器放置位置非常重要，下面是常见的两种 Sensor 安装位置：

（1）传感器放在防火墙外

如果放置在防火墙之外，那么它允许最大范围的监控进出内网的所有流量。传感器能监测到许多防火墙阻断的试探性攻击。但是把传感器推上一线，那么就意味着没有任何保护，很容易受到黑客的袭击。

（2）传感器放置在防火墙内

比较常见的另一种方式就是放在防火墙内部，这样能对成功通过防火墙的流量进行分析和监控，后面我们会详细讨论这种放置方式。

2.2 分布式 OSSIM 体系

当监控千兆网络时，单机安装的 OSSIM 系统会带来性能问题，目前最新的 OSSIM 系统都支持分布式部署。OSSIM 最常见的安装方式是三层体系，分步式部署如图 2-1 所示。

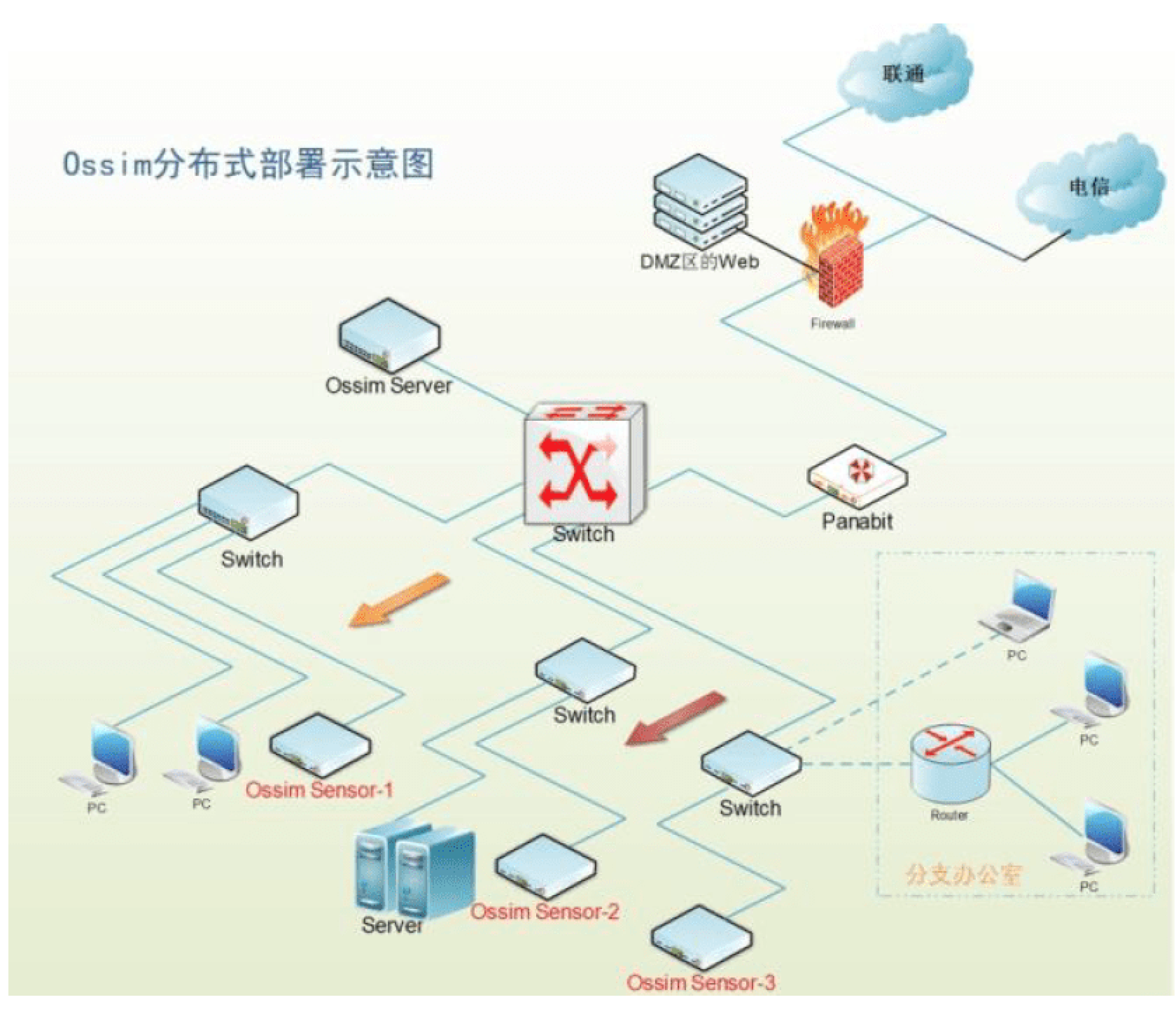


图 2-1 OSSIM 分布式部署示意图

1. 第一层传感器（Sensor）

收集数据包的第一层就是传感器（Sensor）层，它通过对镜像到 SPAN 的流量进行监控来

发现异常和入侵行为。对传感器的要求如下：

- 实现收集所有数据包必须在交换机上做端口镜像（SPAN）；
- 从性能和安全上考虑，OSSIM 传感器遵循分配最小特权的原则；
- 传感器安装两块网卡：一块作为管理接口使用，配有 IP 地址，另一个则为嗅探接口使用，无须配置 IP 地址，嗅探器程序监听在该接口，这种配置是可管理性和安全性的最佳组合，设置思路是让抓包（数据包的截获与分析）的数据包从一个接口进入，所有报警从另一个接口送出。

抓包的网卡不用分配 IP 地址，直接连接到受监控网段，以便流量以一个方向流通，也就是从受监控网段流向传感器。正因为接收数据包的网卡没有 IP，从而也避免了黑客直接攻击传感器。注意，管理接口需要分配 IP 地址，与监控网段不在同一个 VLAN。

2. 第二层服务器（Server）

服务器层主要是收集传感器收集报警数据，并将其转换成用户便于理解的格式。报警数据经处理后被导入 MySQL 数据库。在 OSSIM 系统里，可同时将 Snort 报警发往数据库和系统日志 Syslog 中。报警存入数据库中便于查询管理，可以更好地管理报警消息，方便 GUI 图形界面为用户展现数据。

3. 第三层 SIEM 分析控制台

第三层是 OSSIM 系统的展现层，是安全分析人员最常用的 Web 前端界面。

2.2.1 特别应用

目前 OSSIM 64 位版本对于监控 10/100M 网络环境没有问题，如果是千兆网络环境，在三层交换机上监控多个 VLAN 的流量显得有些吃力，交换机设置端口镜像后同样会使 CPU 占用率上升。据统计，当流量超过 500Mb/s 时，在 OSSIM 分析数据包时会出现响应迟缓，伴随丢包现象出现，而且准确性迅速下降。

其实在 IDS 系统中，存在数据处理瓶颈。数据包处理的难点是 IP 碎片重组和 TCP 流量重组。而这两种重组过程在 IDS 系统中完成，需要对数据包进行深度检测（例如进行数据包模式匹配）。在千兆网络环境中，IP 碎片重组和 TCP 流重组需要强大的 CPU 处理能力，比如 TCP 用于控制数据流的窗口，最大可以为 64KB，当 IDS 系统需要同时维护 1,000,000 个 TCP 会话时，即使缓冲区大小为 4KB，其所占的内存空间达到 4GB。面对复杂的过程，系统响应速度会有所下降。

2.2.2 多 IDS 系统应用

有些特殊场合为了鉴别各种 IDS 的特性，会同时使用多套 IDS 系统，如安装 OSSIM 然后同时使用 HP-Arcsight 分析网络数据包（或 IBM QRadar），这时传统端口镜像（SPAN）的方

法无法满足。也就是说遇到需要将一个监测数据流传送给多台监测设备的情况，采用 SPAN 无法满足。

在网络核心设备上打开 SPAN 需慎重，如果 CPU 利用率长期超过 20%，则不建议打开，以免影响网络性能。

对于这两种情况我们可以采用网络分流器的设备解决，它是一个独立的硬件，支持千兆甚至是万兆网络环境（比如 Gigamon 的方案），而且它不会对已有网络设备的负载带来任何影响，这与端口镜像等方式相比具有极大的优势。它的分流模式是将被监控的 UTP 链路用 TAP 分流设备一分为二，分流出来的数据接入采集接口，为信息安全监控系统采集数据。

2.3 安装前的准备工作

工欲善其事，必先利其器。作为 OSSIM 的使用者，对于企业网中部署 OSSIM 你真的准备好了吗？从软件方面看对于系统维护、网络管理以及安全管理知识体系是否全面了解呢？在本章中部署 OSSIM 是需要使用者具有系统工程师、网络架构师和安全分析师多种角色的知识，下面重点从硬件选型上讲解 OSSIM 准备安装前的注意事项。

2.3.1 软硬件配备

（1）首先确定监控范围。需要监控几个网段内的多少台服务器，每台设备的日最高流量为多大（需要按峰值考虑），每台设备都需要能联系到相应的管理员。

（2）确定监控对象，虽说 OSSIM 能够监控多种设备，但实际上为了保证性能，不能无节制地打开各种服务。

（3）从人员配备上看，需由专人负责管理，维护 OSSIM 的人员，首先应该是具有一定工作经验的 Linux 工程师，熟悉 Linux 系统+网络架构+MySQL+PHP，即熟悉 Linux 系统运维、MySQL 数据库运维、信息安全管理，也需要掌握网络编程知识。

（4）硬件选择，可以采用品牌服务器，对于中小企业也可以根据自己需求，以 OSSIM 4.8 系统为例，目前系统对多核性能支持得比较好，推荐采用至强 E 系列处理器，OSSIM 在漏洞扫描、Ossec 扫描、Snort 事件分析时会消耗大量 CPU，所以要尽量选择高性能 CPU，尤其是在 OSSIM USM 发展到 5.0 之后，数据库采用了 MySQL 5.6，对多 CPU 处理能力需求更高。

就内存而言，只有一个道理，越大越好。当数据库的全部数据页能保存在缓冲池中，那么其性能理论上是最优状态。对于新版本 OSSIM，建议需要配备 16GB 以上内存，经过长期测试，对于 OSSIM 4.3（64 位）版本系统而言，如果内存分配小于 6GB，在实际测试中系统工作一段时间之后，由于内存溢出等问题，可能出现某些服务自动重启或没有响应的情况。

所以 32GB 内存是系统稳定运行的经验值（而且监控选项和插件选项需要针对性地打开），

另外系统还需要 2TB 的存储空间，有条件内存配备 32GB 以上比较理想。笔者在测试环境中采用自己攒的服务器，配置为：华硕 P8Z87-K+Intel I7 4770K+32G 内存+双千兆 Intel 网卡+4T 硬盘，这个配置下安装 OSSIM 4.8 一次性通过，运行效果比较理想。

在持久存储上通常使用多块硬盘组成 RAID 阵列，OSSIM 系统中常采用 RAID 1+0 模式，但机械磁盘本身的特性决定了其 IOPS 性能比较低，而通过多块盘做 RAID 虽能提升 IOPS，但对于 OSSIM 系统而言依然缓慢，所以对于有条件的企业建议采用固态硬盘，当前 SSD 能轻松达到 50000。

固态硬盘可分为 PCIe 和 SAS（SATA）接口。PCIe 有着最好的性能，但价格较贵。而 SAS 接口的一个好处是易于安装，升级当前服务器的存储到 SAS 接口的固态硬盘仅需拆卸原来的机械硬盘即可。而 PCIe 需要拆开服务器的背板，工程量较大，普通系统工程师恐难胜任。

（5）对于 Broadcom Netxtreme 网卡所遇到的问题

市面上有一些 HP 和 Dell 的服务器采用集成 Broadcom Netxtreme 网卡，安装 OSSIM 系统就会遇到找不到网卡驱动的问题，因为 Debian 系统无法加载 firemware bnx2 模块，这时，需要到 Broadcom 官网下载 For Debian 的驱动，然后通过 U 盘安装 。

成功加载驱动后，在系统内就能查看到详细信息：

```
# dmesg | grep bnx2
[    1.909228] Broadcom NetXtreme II Gigabit Ethernet Driver bnx2 v1.7.5
[    2.634060] firmware: requesting bnx2-06-4.0.5.fw
[    3.185810] firmware: requesting bnx2-06-4.0.5.fw
```

新版本的 OSSIM 在这方面进行了改进，虽增加了 firmware-bnx2 包，但没能够对 Broadcom Netxtreme 系列网卡有所改善，普通用户推荐 Intel Pro 100/1000 系列服务器网卡（对应芯片型号 82558/82559）。还有更多 Debian 支持的 PCI 设备可到 http://wiki.debian.org/DeviceDatabase/PCI 中查询。

对于 OSSIM 服务器的数据存储问题，可使用已有的存储系统。网卡方面选用 Intel 的双千兆网卡比较合适，另外在交换设备上做好 SPAN 设置这一步至关重要，详细操作后面会讲到。需要将流量镜像到 Sensor 的网络接口。

2.3.2 传感器部署

单机模式下 OSSIM 系统放置在什么位置比较合适？有两种方案：

- 部署在公司互联网的出口（主要抓取和分析上网流量）。
- 部署在 DMZ 对此区域的服务器监控，如图 2-2 所示。

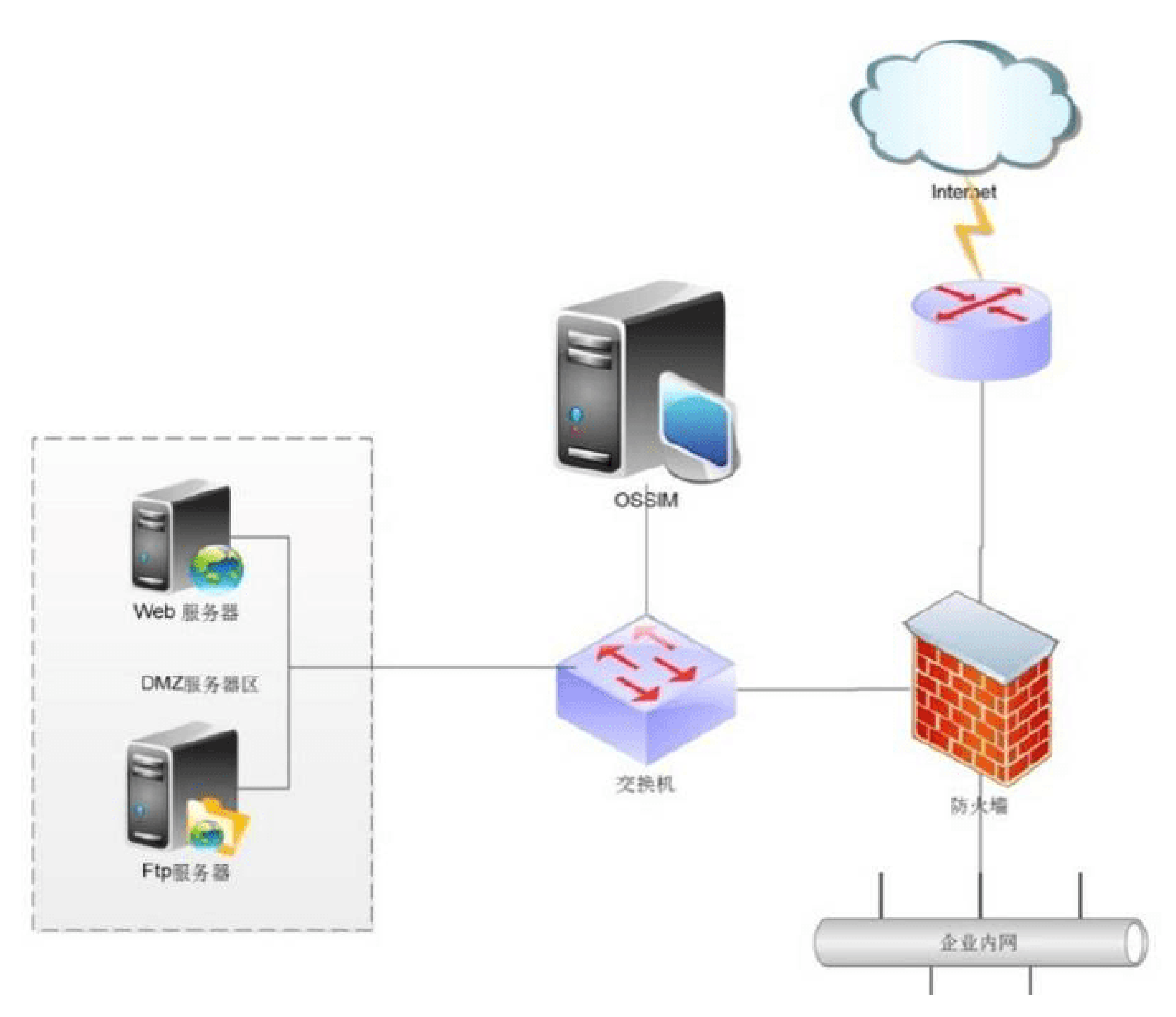


图 2-2　单台 OSSIM 系统安装部署

单台 OSSIM 系统安装非常方便，按提示操作即可。

采用分布式监控，部署在重点 VLAN，有选择地监控机器设备，如图 2-3 所示。

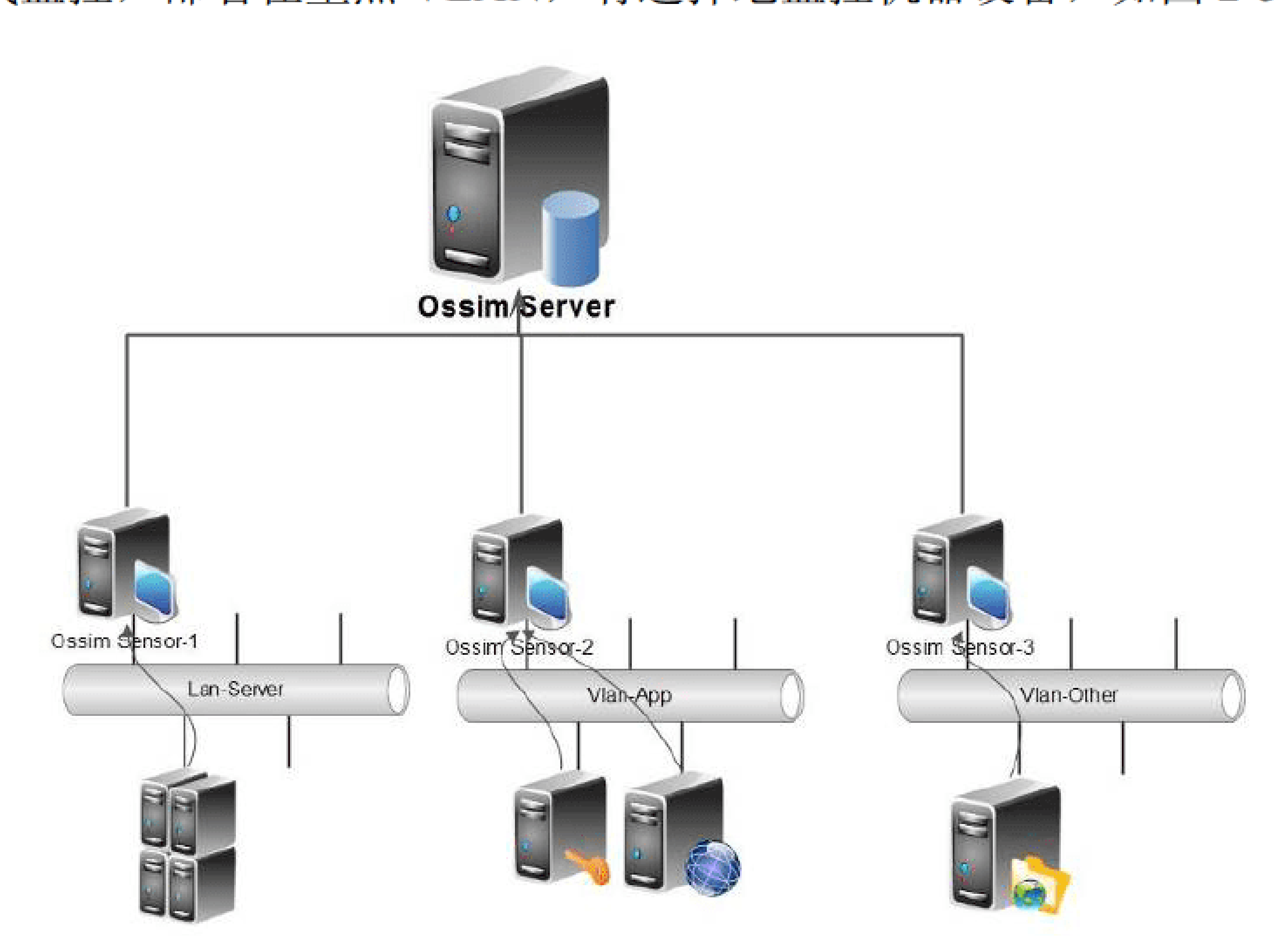


图 2-3　分布式流量监控部署示意

分布式系统架构 C/S 模式，整体来讲由代理和服务器所组成，图 2-3 中的 OSSIM sensor-1、sensor-2、sensor-3 这 3 台机器为代理，OSSIM Server 为服务器，这里的代理机器上包括所选择的监控插件，以及探测器（sensor），主要通过插件来捕获当前 VLAN 中需要监控的数据，经过代理机器上的 Sensor 分析形成 OSSIM Sever 服务器能够读取的日志，发送给 Server 进行关联处理后，统一存放到事件数据库中，这样做的目的是为 OSSIM 的审计模块提供关联分析

和风险评估的数据。

2.3.3 分布式 OSSIM 系统探针布局

大型网络环境中 OSSIM 通常采用 5 个探针来进行检测。探针 1、2 所在的网段的数据流量是最大的，为有效防止误报（对无害的网络行为发出的告警）和漏报，设置了两个探针。

- 探针 1 位于防火墙外，可以查看所有来自 Internet 的攻击。为了减少受攻击风险，探针 1 使用无 IP 地址的网卡进行监听，以保证网络入侵检测系统自身的安全，通过另一块网卡接入内网并为其分配内网所使用的私有地址，以便从内网访问分析控制台程序 ACID。
- 探针 2 位于防火墙内部，使用交换机（SPAN 口），任何其他端口的进出数据都可从此得到，不过采用此端口可能降低交换机的性能。该探针可以看到外部所有突破防火墙对内网的攻击，发现防火墙的设置失误和漏洞所在，还可以发现一些由内网发起向外网服务器的攻击。
- 探针 3 位于服务器组内，用于检测所有对各服务器的攻击。
- 探针 4 位于工作站组内，用于检测所有对各工作站的攻击。
- 探针 5 位于内部办公网，用于检测所有对内部办公网的攻击。

这 5 个探针整体布局如图 2-4 所示。

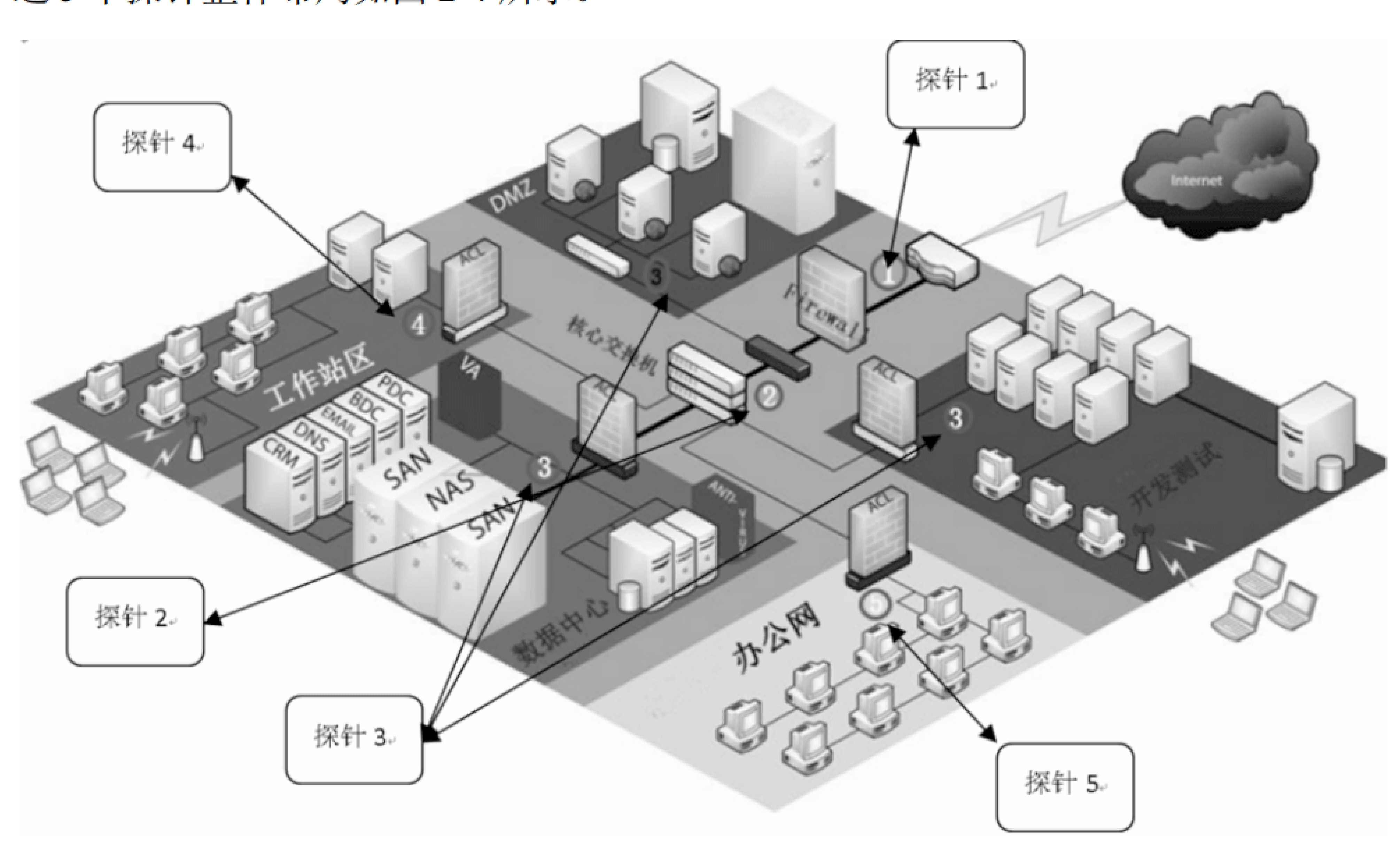


图 2-4　OSSIM 探针位置

提示：分布式网络中各探针分布在各个 VLAN，这时必须保证时间精准，这主要通过 NTP 服务实现。

2.3.4　OSSIM 服务器的选择

部署 OSSIM 服务器时常会遇到两类问题，一类是无法识别硬盘，另一类是无法识别网卡。对于 Dell、HP 和 IBM 品牌 X86 服务器系列，官方默认对 Windows 以及 Linux 发行版 Rad Hat、SUSE Linux 提供 RPM 格式的驱动支持，它们只提供 Red Hat 和 SUSE 的硬件兼容列表，对于 Debian Linux 平台支持相对较差。

一些使用 OSSIM 的用户，为服务器 Raid 卡安装驱动头痛不已。经测试 HP ProLiant DL160 G6 、DL360、DL380（G8）、IBM X3100M4 以及方正圆明 LT200 2600 等服务器都能顺利安装 OSSIM 4。大家在选择一款专业服务器时，需要确认它是否支持 Debian Linux 系统。

OSSIM 是基于 Debian Linux 的系统，所以并没有包含最新服务器的网卡驱动和 Raid 卡驱动，在厂家那里没有提供兼容列表时，大家可以在 http://kmuto.jp/debian/hcl 上查询机器是否适合安装。例如查询 IBM X3650 机器是否能安装就可以查询 http://kmuto.jp/debian/hcl/IBM/x3650/。

在网卡的选择方面大家需要注意，有条件的部门可以选择带队列功能的网卡，例如 Intel 82576 千兆网卡，它支持 PCIe 2.0 X 4，支持 MSI-x 中断，支持 8 个 RSS 队列。

对于 CPU 的选择，尽量选择多路至强处理器，因为在 OSSIM 框架内的绝大多数服务都支持多线程，如表 2-1 所示。

表 2-1　OSSIM 中的多线程与单线程服务

支持多线程服务	MySQL、RabbitMQ、Suricata、Nmap、Memcached、ossim-framework、ossim-server、Ntop、Nagios、Apache、OpenVas
支持单线程服务	Snort、Squid、Redis、iptables

在多核运算上，不是 CPU 越多越好，比如我们选择一款 Xeon E5-2680 CPU，8 核心 16 线程 20MB 三级缓存强大处理器。如果在增加 CPU 对提升系统性能并不明显时，我们也需要知道如何查看 CPU 指标，这个指标主要是通过在系统中查看/proc/cpuinfo 文件获取，我们会发现以下信息：

- CPU 的物理个数。
- 具有几个逻辑核。
- CPU 是否启用超线程。
- CPU 的主频。
- 逻辑 CPU、CPU 型号。

对于选型安装我们再看个例子，IBM X3650 7979 服务器安装 OSSIM，由于 Debian 对 Broadcom NetXtremeII 网卡不支持（笔者在 Dell PowerEdge R720 服务器上部署时同样遇到过这样的问题），所以如果在 X3650 服务器上安装 OSSIM 系统会出现找不到网卡的情况。我们首先到 Debian 官网下载驱动，然后复制到 U 盘，再通过 U 盘加载到 X3650 服务器上并安装，最后重启系统。

下载地址：http://packages.debian.org/sid/all/firmware-bnx2/download。安装如下：

```
#dpkg -i firmware-bnx2_0.40_all.deb
```

在系统引导时可以看到类似“load firmware file bnx2-06-4.0.5.fw”的信息，表示加载成功。然后我们就可以通过 ossim-setup 开始为服务器配置 IP。

Linux 系统中把常用应急驱动都封装到 initrd.img（或 initrd.gz）内核中，通常来讲 Linux 版本越高内核体积越大，相应支持的硬件驱动也就越多，表 2-2 中总结了常见 Linux 发行版的内核容量。

表 2-2　主要服务器 Linux 版本内核对比

主要发行版	版 本	initrd.img 容量
Redhat 企业版	5.0 版	4.9MB
	5.5	7MB
	6.0	28MB
	6.2	36MB
	7.0	37MB
Suse 企业版	10 sp2	10MB
	11	21MB
	12	23MB
Debian	5.0	4.4MB
	6.0	16MB
	7.0	23MB

如果 Raid 卡或网卡无法加载驱动，这时就需编译成可加载模块来安装。有时在安装 OSSIM 过程中总会找不到一些硬件，这里提供一个方法，使用 Grml64 光盘，它是基于 Debian 的系统，它里面包含了许多工具，可以直接用光盘启动。Grml 提供更多自动硬件识别，它主要能帮助识别服务器的硬件设备的具体型号（可以自行下载相应的驱动，无须开机箱盖查看硬件）。下载地址为 http://grml.org/download/，其完整版容量为 350MB。

2.3.5　网卡的选择

通常，大家在实体服务器上通过光盘安装 OSSIM 过程中，没有提示输入 IP、网关等配置，进入系统后才发现网卡没有加载驱动，此时你再返回去下载服务器网卡驱动比较麻烦，那到底 OSSIM 系统需要什么样的网卡呢？如果 OSSIM 工作在千兆网络环境，建议加装一块性能优异的网卡，首推 Intel Pro 网卡，它是著名品牌且性能稳定，可显著地改善服务器网络性能的特性，解决网络传输瓶颈。

哪种网卡最适合 OSSIM 呢？从安装方便程度和价格上看当属 Intel Pro 10/100/1000 网卡，但它的吞吐量并不是最好，OSSIM 自带 Intel Pro 网卡驱动，另外选择 Realtek 瑞昱 8169 芯片（OSSIM 直接带驱动）网卡也是一种选择，它比 Intel 略逊一筹，比它更好的例如 Intel Gigabit ET Quad Port Server Adapter，型号是 E1G44ET，这需要你手动安装驱动，这块基

于两个 82576 芯片的强大四口千兆网卡，适合大流量网络环境下监控，但价格比较贵。

2.3.6　手动加载网卡驱动

Intel 网卡下载地址：https://downloadcenter.intel.com/zh-cn，进入网页后在查找类别中选择“以太网控制器”，然后选择对应芯片型号，例如 82574 千兆以太网控制器。在弹出页面中选择操作系统为 Linux，在下载驱动源码包中需要注意内核版本号，OSSIM 的 Linux 内核版本为 2.6.32，所以需选择 2.6 内核的网卡驱动。手动编译源码驱动步骤如下：

（1）检查 GCC 编译环境，若没有则需要安装。

（2）安装 Kernel 2.6.32 源码包，将内核文件 linux-2.6.32.tar.bz2 解压到/usr/src 目录下。

（3）安装所需软件包：

```
#apt-get install kernel-package  libncurses5-def  fakeroot
```

（4）驱动程序编译及安装，例如从官网获取 Intele1000e 的网卡驱动程序源码包为 e100e-3.0.tar.gz，将驱动解压后复制到/usr/src 目录。在 src 目录下依次执行 make（编译驱动程序源码）、make install（安装相应的驱动程序），安装完毕以后将驱动程序生成的*.o 复制到/lib/modules/内核版本/kernel/drives/net 目录下，然后执行 depmod -a 加载驱动程序。

（5）驱动程序测试：

```
#modprobe  e1000e                \\ modprobe 命令加载网卡模块
```

使用 lshw -c network 列出有关网卡的详细信息，得到的信息中包括版本+NAPI 信息表示成功安装。

2.3.7　采用多核还是单核 CPU

在 OSSIM 中集成了很多优秀的抓包工具，例如 tcpdump、snort/suricata。这些工具都具有数据包捕获函数库，例如 PF_Ring，它们都是以库函数为基础的软件方式抓包，在老版本中采用 Libpcap 抓包，由于它接收数据包时产生的中断开销，以及将接收到的数据包从网卡复制到内核，再从内核复制到用户空间消耗大量 CPU 资源，所以不适用高速链路。若提高抓包效率就必须减少内存复制次数，改变中断方式，减少不必要的 CPU 中断，PF_Ring 机制在这种需求下诞生。在 OSSIM 中采用了 PF_Ring+NAPI 的捕包机制。

对于一个流量监控模块来说，必须要求足够的 CPU 资源，来对捕获的数据包做深一层处理和分析，否则捕获的数据包会被丢弃。因此有必要测试在各种数据包大小下，包捕获有没有发生丢弃，注意观察 CPU 使用率。

即使使用了 Suricata 支持多线程，在多核平台上抓包，也没有成倍提升性能，可以说采用多核和 Suricata 后抓包效率比过去提升了不少，但还有一部分不可避免的 CPU 消耗主要集中在内核空间。

2.3.8 查找硬件信息

有关硬件的信息包括 PCI 总线设备、USB 设备、SCSI 设备等相应的设备信息至关重要，下面这几条命令需要掌握：

lspci 命令会列出系统中的 PCI 总线设备，使用参数-v、-vvv 可以得到更多详细信息。

```
#lshw                        \\硬件配置信息检测工具
```

USB 设备的信息可以用 lsusb 命令获得。如果希望查看硬盘剩余空间就要借助于 df 命令。

```
#df -h
```

查看 CPU 型号及内核数。

```
#/proc/cpuinfo
```

2.3.9 OSSIM USM 和 Sensor 安装模式的区别

初学者在安装 OSSIM 时，首要问题就是选择 USM 安装还是 Sensor 安装，之所以产生这样的疑问是因为对于这两种安装模式中的组成不了解。OSSIM USM 包括 Sensor 的所有模块，为什么要单独分出来 Sensor 安装方式？这是为了分布式部署 OSSIM 时用，如果用户选择在单台服务器上混合部署，那么就选择 USM，如果你希望在多个 VLAN 分别部署，则除了安装 USM 之外还需要在各个嗅探点安装 Sensor，将 Sensor 和 Server 连接。

USM 和 Sensor 主要区别就在所安装模块上。下面参考 OSSIM USM 4.8，更新版本中的主要模块类似表 2-3，系统对应服务组件区别也如表 2-3 所示。

表 2-3 Alienvault OSSIM USM & Sensor 主要模块对比

组件/服务分类	模 块	OSSIM USM	Sensor
Alienvault	Alienvault-agent-generator、alienvault-api-core、alienvault-api-script、alienvault-api-center、alienvault-doctor、alienvault-dummy-sensor、alienvault-logrotate、alienvault-monit、alienvault-openvas、alienvault-openvas-plugins、alienvault-plugins、alienvault-rsyslog	√	√
	Alienvault-apache2、alienvault-api、alienvault-crosscor、alienvault-directiv、alienvault-dummy-da、alienvault-dummy-fr、alienvault-dummy-se（server）、alienvault-framewor、alienvault-idm、alienvault-memcache、alienvault-mysql、alienvault-php5、alienvault-postfix	√	×
Apache	Apache2、apache2-doc、apache2-mpm-prefork、apache2-utils、apache2-bin、apache2-common	√	√
Netflow	Fprobe、fprobe-ng	√	√
	Nfdump、nfsen	√	×
Memcachd	memcached	√	×

（续表）

组件/服务分类	模 块	OSSIM USM	Sensor
Mrtg	mrtg	√	√
Munin	Munin、munin-common、munin-node	√	√
Monit	Alienvault-monit、monit	√	√
Nagios	Nagios-images、nagios-plugins、nagios-plugins-basi、nagios-plugins-stan、nagios3、nagios3-common、nagios3-core	√	×
Ncurses	ncurses-base、ncurses-bin、libncurses5、libncursesw5	√	√
Ocsinventory	Ocsinventory-agent、ocsinventory-server	√	×
Ntop	Alienvault-ntop、ntop、pfring	√	√
OpenSSH	Openssh-blacklist、openssh-client、openssh-server	√	√
OpenSSL	Openssl-blacklist、openssl	√	√
OpenVPN	Openvpn、openvpn-blacklist	√	√
Ossec	Ossec-hids	√	√
OpenVAS	Openvas-administra、openvas-manager、openvas-scanner、openvas-cli	√	√
MySQL	Percona-server-clie、percona-server-comm、percona-server-serv、mytop	√	×
Nagios	Nagios3-common、Nagios3-core、Nagios-plugins	√	×
Postfix	Postfix	√	×
Rsyslog	rsyslog	√	√
Samba	Samba-common、smbclient	√	√
Snort	Snort、snort-common、snort-common-librar、snort-rules-default	√	√
Suricata	Suricata、suricata-rules-default	√	√
Squid	Squid、squid-common、squid-langpack、squid3、squid3-common	√	×
Snmp	Snmp、snmpd	√	√
Prads	Alienvault-prads、prads	√	√
OSSIM	OSSIM-agent、ossim-cd-configs、osism-cd-tools、ossim-contrib、ossim-database-migration、ossim-downloads、ossim-geoip、ossim-menu-setup、ossim-repo-key、ossim-utils	√	√
	OSSIM-compliance、ossim-framework、ossim-gramework-dae、ossim-mysql、ossim-mysql-ext、ossim-server、ossim-taxonomy	√	×
OCS Inevent Server	ocsinventory-server agent	√	×
Redis	Redis Server	√	×
HA	Cluster-agents、cluster-glue、heartbeat	√	√
Capturing Packet	Wireshark-common、tshark、dsniff、tcpdump、libpcap、pfring、tcpreplay	√	√
Rabbitmq	Rabbitmq-server	√	×
Rsyslog	Alienvault-rsyslog、rsyslog	√	√
Erlang	Erlang（用于分布式系统中结构化的编程语言）	√	×
C 编译环境	gcc-4.4-base、libgcc1	√	√
	gcc-4.4	√	×
Perl	Perl、perl-base、perl-modules	√	√

（续表）

组件/服务分类	模 块	OSSIM USM	Sensor
PHP	Php5、phpgd、php5-adodb、php5-cgi、php5-common、php5-mysql、php5-odbc、php5-snmp	√	√
	Php-db、Php-soap、php-xajax、php5-memcache	√	×
Python	Python	√	√
Tcl	Tcl8.5、tcl8.5-dev	√	√
Scan Tool	nikto	√	×
	nmap	√	√
Iptables	iptables	√	√
随机密码生成	Pwgen	√	√

2.3.10 OSSIM 商业版和免费版比较

OSSIM 免费版中一些功能受到限制，例如 Logger、Sing、Forward Alarms、Forward Events 四项服务禁用。服务器组件查询位置：Configuration→Deployment→Components→Servers，只输出基本报表。在表 2-4 中总结了免费 OSSIM 和商业版 OSSIM USM 的主要区别。

表 2-4　商业版和免费版的区别

	OSSIM	USM All-in-One	USM Standard&Enterprprise
适应人群	安全研究人员	中小型企业	大型企业和组织
价格	Open Source	3600$	-
应用案例	网络流量、协议分析 网络资产管理、漏洞扫描 安全事件管理 可用性管理、完整性检查 报告输出（Alarm、Asset、PCI-DSS 等）	比开源版增加了合规管理和报告（PCI、HIPAA、ISO 27002（SOX））、事件响应、日志管理	
技术支持	社区	厂商支持	
规则/插件数量	少量	规则丰富可持续更新	
管理	集中管理和配置	集中管理和配置	
威胁情报	社区的开发组织	Alienvault 实验室威胁智能订阅（每周更新）	
报 告	社区提供的基本功能	提供数千种威胁报告方便导出多种格式	
访问控制	基于角色的访问控制权限	基于角色的访问控制权限以及丰富的模板	
可扩展性&性能	不易扩展性，能受 MySQL 限制	可在线提供扩展支持，包括单机部署和多机分布部署，便于扩展服务器和 MySQL	
Alienvault Logger	无	有	
转发 Alarm 和 Event	不能	能	
智能事件分析	无	有	
SIEM 数据备份数量	少	多	

OSSIM USM ALL-IN-ONE 包含 Sensor、Framework、Server、Database。

2.3.11 OSSIM 实施特点

OSSIM 系统的实施过程中，各个单位需要结合自身的实际情况进行，以下是作者总结的几个关键因素。

（1）对企业内的业务系统进行调研，主要目的是识别信息系统的业务种类和特性，了解具体信息系统处理的有多少种业务应用，这些业务应用各自的业务内容、业务流程、业务用户和管理用户等，从中明确网络系统的业务特性，这样有助于根据业务特征和安全需求，制定安全策略，实际上也是帮助安全人员了解业务系统的网络架构，此过程中要能调查资产情况。

（2）选择的 OSSIM 版本要具有良好的网络环境适应性，不能因为新上系统而大幅调整网络结构，所以采用旁路接入方式进行嗅探是比较好的方式。

（3）支持快速部署，具有良好的用户体验，OSSIM 最大亮点是能够让各类用户接受，能够快速部署，并且在部署时具有友好的 Web 引导界面。

（4）具有较好的可扩展性，能够提供不断更新的安全检查引擎和规则库升级。

（5）总体规划、分步实施。

“总体规划、分步实施”是 OSSIM 系统实施的重要策略。在总体规划的同时，将系统的建设目标分解为若干可操控的项目，分步实施，例如安装 OSSIM USM 之后，首先升级系统，然后备份初始配置，然后开始逐一安装 Sensor，调试一个加入一个，切忌同时加入了多个 Sensor 后，再统一联调，此时出现了错误就不便于排查。

（6）协调好技术与管理的问题。OSSIM 实施涉及技术、人员和流程，其中人是最重要的、最不容易搞定的因素。为了获得最终的分析效果，需要收集各种门类的信息，但不幸的是，这些信息往往隶属于不同的部门，例如网络和主机，还有应用可能分属于几个部门。如何统一收集和分析这些信息，同时确保不会引发大家的担忧是必须考虑的问题。

（7）OSSIM 规划设计需要有前瞻性，而且在公司拥有 OSSIM 系统之前，必须配置一个安全团队，公司在组建安全团队时，至少需要 1 名安全行业多年经验的专家作为技术带头人，能够将他们的经验转换为知识。这个 SIEM 专家需要很高的素质，需要善于调配每种服务器上收集到的数据，需要考虑数据如何存放，以及 SIEM 系统如何分类等等。如果企业没有 SIEM 专家和安全团队，贸然上 OSSIM 系统则很难成功。

（8）OSSIM 上线要素。为了实现 OSSIM 管理，把系统上线远远不够，OSSIM 建设管理是一个慢工细活，需要长期积累和沉淀。有了这套东西，在新一代攻击手段面前依然十分脆弱，仍然会有新的问题产生。如何才能将网络安全防线真正从网络边缘扩展落实到所有网络终端节点上呢？从广大承建方及使用方看来，后续跟进的合理化使用管理能帮助用户实现这一点。当

系统上线以后，要做到全网无盲点的 OSSIM 管理，都不是一步到位的，需要将管理制度和流程通过循序渐进的过程逐步建立起来，并进行不断地完善优化。OSSIM 的部署几乎跨越整个机房设备，从网络基础设施到网络管理，所以 OSSIM 的涉及面非常广泛，包括系统、数据库、中间件、存储、网络基础设施等诸多环节。

2.3.12 OSSIM 管理员分工

建设一套合格的 OSSIM 系统，不仅是从技术角度去完善技术产品，一方面还需要一个完整的系统运行管理流程，建立一个有技术能力、效率高的管理团队。该管理团队由单位领导牵头，以信息安全管理部门人员为主体，配合与系统架构师、系统管理员、网络管理员、数据库管理员相关人员组成，系统运行的管理、操作、审计等权限明确，做到三权分立。下面就 OSSIM 系统建设后管理人员分工建议如下：

（1）安全事件分析人员

通过筛选和分析 SIEM 收集的安全事件，发现安全攻击与威胁，负责监视安全事件告警的 TopN、安全事件的分类、报告安全事件最多的主机 Top 10、每天/每周安全事件变化趋势、病毒报告 Top 10、详细的安全事件告警（根据收到的告警再修改关联指令策略）、SIEM 事件控制台、管理资产的变化，漏洞扫描报告，追踪漏洞修复情况、网络服务可用性、流量监控和协议分析、定期收集系统各项报告，通过各种报表和数据分析出网络中存在的病毒感染、外部入侵、信息泄露、员工违规等异常事件。

（2）OSSIM 系统架构人员

基于业务重要性和需求的变化，不断地优化安全事件和流程，并负责企业日志容量估算，系统架构规划、安装、系统调优、配置备份、数据库优化。

（3）安全审计人员

安全审计人员需要具有一定的 ISO27001、PCI-DSS 以及等级保护的知识，主要负责评估整个系统的运行情况，判断管理工作的合理性、合规性，做到使系统运行更符合实现具体目标。

2.4 混合服务器/传感器安装模式

OSSIM 的安装工具遵循经典的 GNU/Linux 安装流程，同时提供了人性化的向导，智能化检测和配置功能。下面我们动手实践一下吧，首先我们选用 OSSIM 的 ISO 镜像（或光盘）安装。

2.4.1 安装前的准备工作

- 需要收集系统信息，对硬盘进行检查、分区及规划。
- 选择安装介质。

大家可以到 http://www.alienvault.com/free-downloads-services 下载 OSSIM 安装包，目前官

网提供 64 位镜像的安装文件。

2.4.2　开始安装 OSSIM

对于初学者而言可以采用混合服务器/传感器模式安装，这种安装模式是将所有 OSSIM 组件都安装到同一台计算机上。接下来我们开始具体操作，由镜像文件（这里以 alienvault-ossim 4.8 ISO 为例）启动系统，安装界面如图 2-5 所示，选择 Install AlienVault USM 4.8（64 Bit）选项，这一选项将包含所有 OSSIM 的 4 个模块（包括 Sensor+OSSIM Server+Database+Framework），接着启动系统。

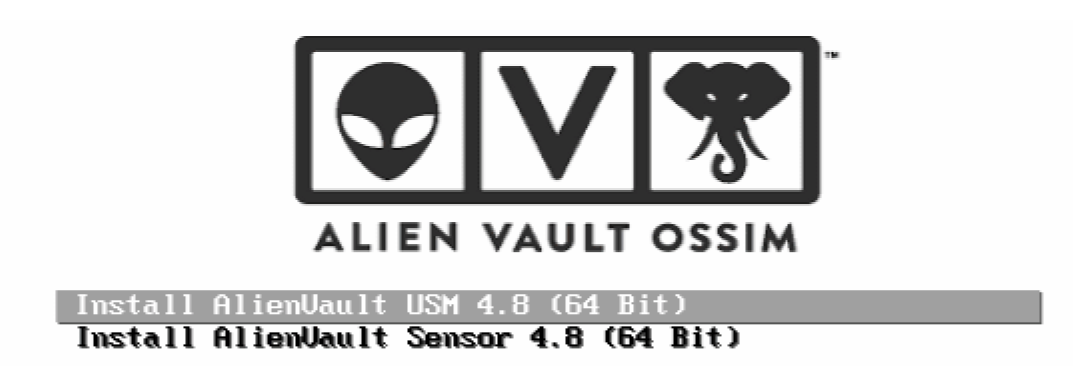


图 2-5　安装界面

启动菜单含义说明如下：

- Install Alienvault USM 4.X（64 Bit）：用混合模式安装完整的 OSSIM。若在这个菜单上按“Tab”键，会进入 Grub 命令行界面，此时如果删除“autoALLinOne”选项，就能进入专家安装模式。这种操作方法参考作者博客 http://chenguang.blog.51cto.com/350944/1723159。
- Install Alienvault Sensor 4.X（64 Bit）：仅安装 Sensor 组件 。同理，若在这个菜单上按“Tab”键，进入 Grub 命令行，删除“autoSensor”选项，即可进入专家安装模式。

对语言与区域进行设置，如图 2-6 所示。

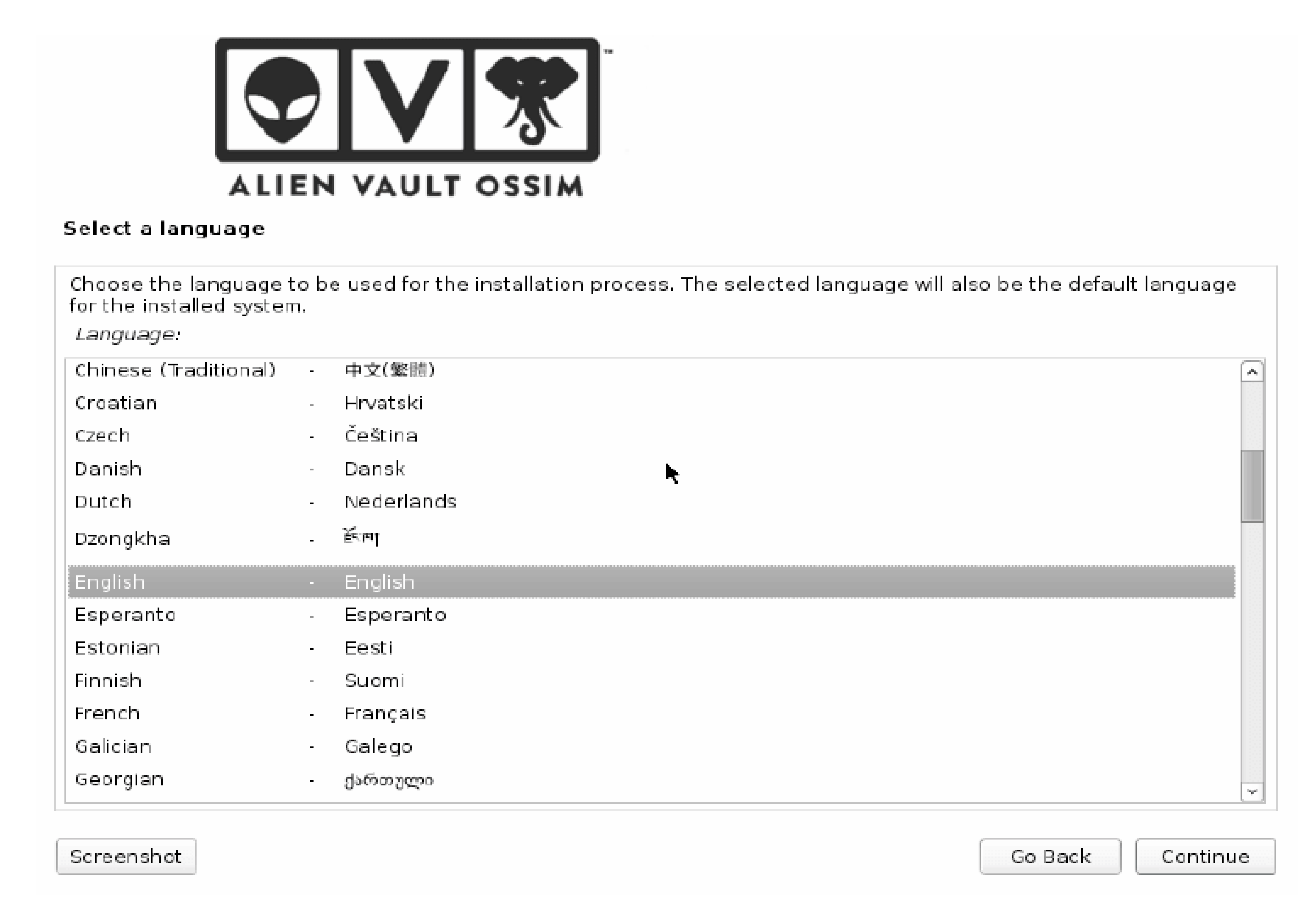


图 2-6　选择语言

OSSIM 的安装界面支持多种语言，一般选择英文。在国家选项中依次选择 Other→Asia→China，如图 2-7 所示。

图 2-7　选择国家

接着选择 United States en_US.UTF-8。接下来选择键盘样式，一般的用户选择美式英语，如图 2-8 所示。

图 2-8　配置键盘

加载完整的安装程序，接下来寻找光盘镜像，中间会提示加载驱动模块，该过程自动完成。注意：在 OSSIM 4.1 之后进行强制分区策略，所以这一步没有用户分区步骤。

接着开始指定 IP 地址。假如在物理服务器上安装，看到如图 2-9 所示界面，说明没有网

卡驱动，可以选择继续安装，待系统安装完毕手动加载网卡驱动。

ALIEN VAULT OSSIM

Detect network hardware

No Ethernet card was detected. If you know the name of the driver needed by your Ethernet card, you can select it from the list.

Driver needed by your Ethernet card:

no ethernet card
3c574_cs: 3Com 3c574 series PCMCIA Ethernet
3c589_cs: 3Com 3c589 series PCMCIA Ethernet
3c59x: 3Com 3c59x/3c9xx PCI Ethernet
8139cp: RealTek RTL-8139C+ series 10/100 PCI Ethernet
8139too: RealTek RTL-8139 Fast Ethernet
8390: National Semiconductor 8390 Ethernet
abyss: Madge Smart 16/4 PCI Mk2 Token Ring
acenic
airo: Cisco/Aironet ISA/PCI 802.11
airo_cs: Cisco/Aironet PCMCIA 802.11
amd8111e: AMD8111 based 10/100 Ethernet
asix

Screenshot　　Go Back　　Continue

图 2-9　未加载网卡驱动

服务器 IP 必须为指定地址（如图 2-10 所示）、网络掩码、网关地址和 DNS 地址。安装过程还需要与时间服务器同步，所以最好能连接外网。如果需要手工修改网络配置，也可以直接修改/etc/network/interfaces 配置文件中的相关参数。配置网络参数完毕后进行时间同步。

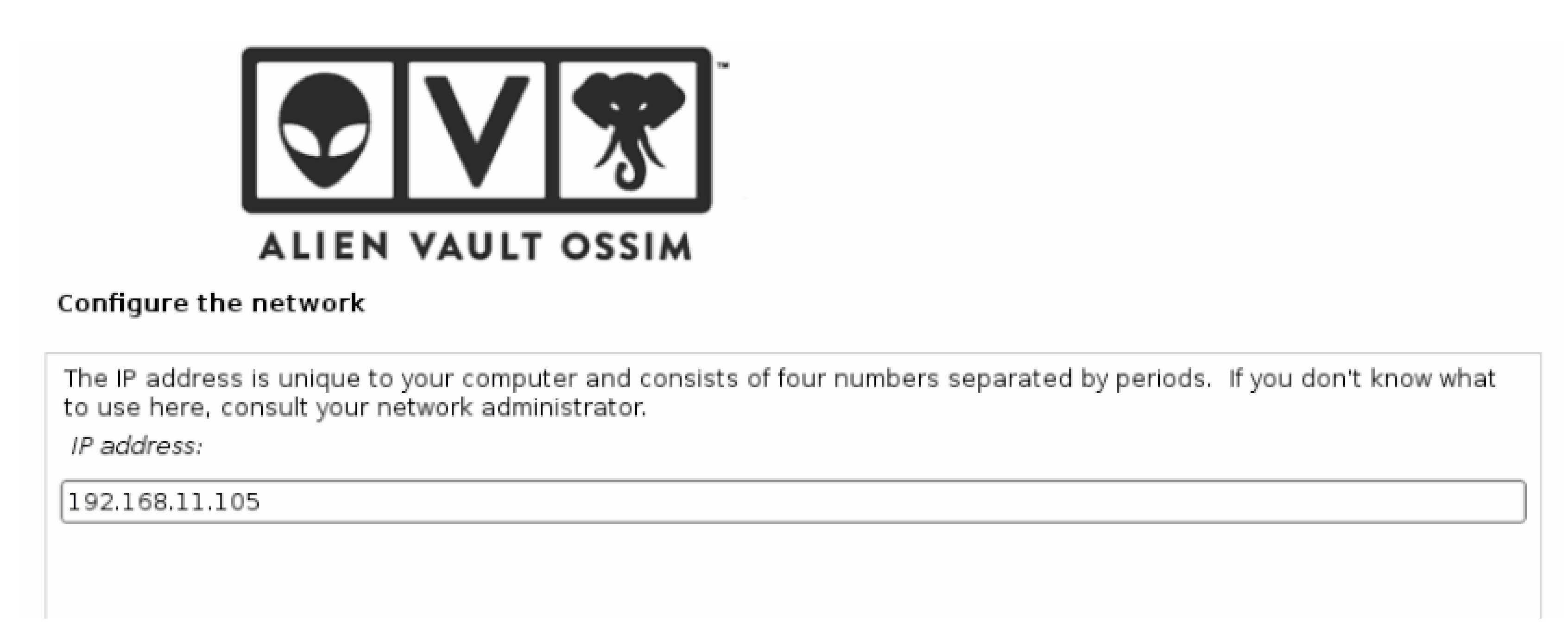


图 2-10　配置 OSSIM IP 地址

接下来几个步骤都无须人工干预，系统将自动完成，整个安装过程大约持续半小时。然后，使用浏览器（支持 IE10、Safari8、Chrome44）登录并进行系统初始化。注意：如果读者选择 Sensor 模式安装，安装步骤和 USM 相同，这里不再赘述。

首次用浏览器登录时，会弹出如图 2-11 的登录界面，需要输入管理员名称（系统管理员用户名约定为 admin）以及密码，在试验环境中修改密码（小写字母和数字的组合），假设为“a1b2c3d4”，然后输入电子邮件地址，公司名称和地理位置是可选项。

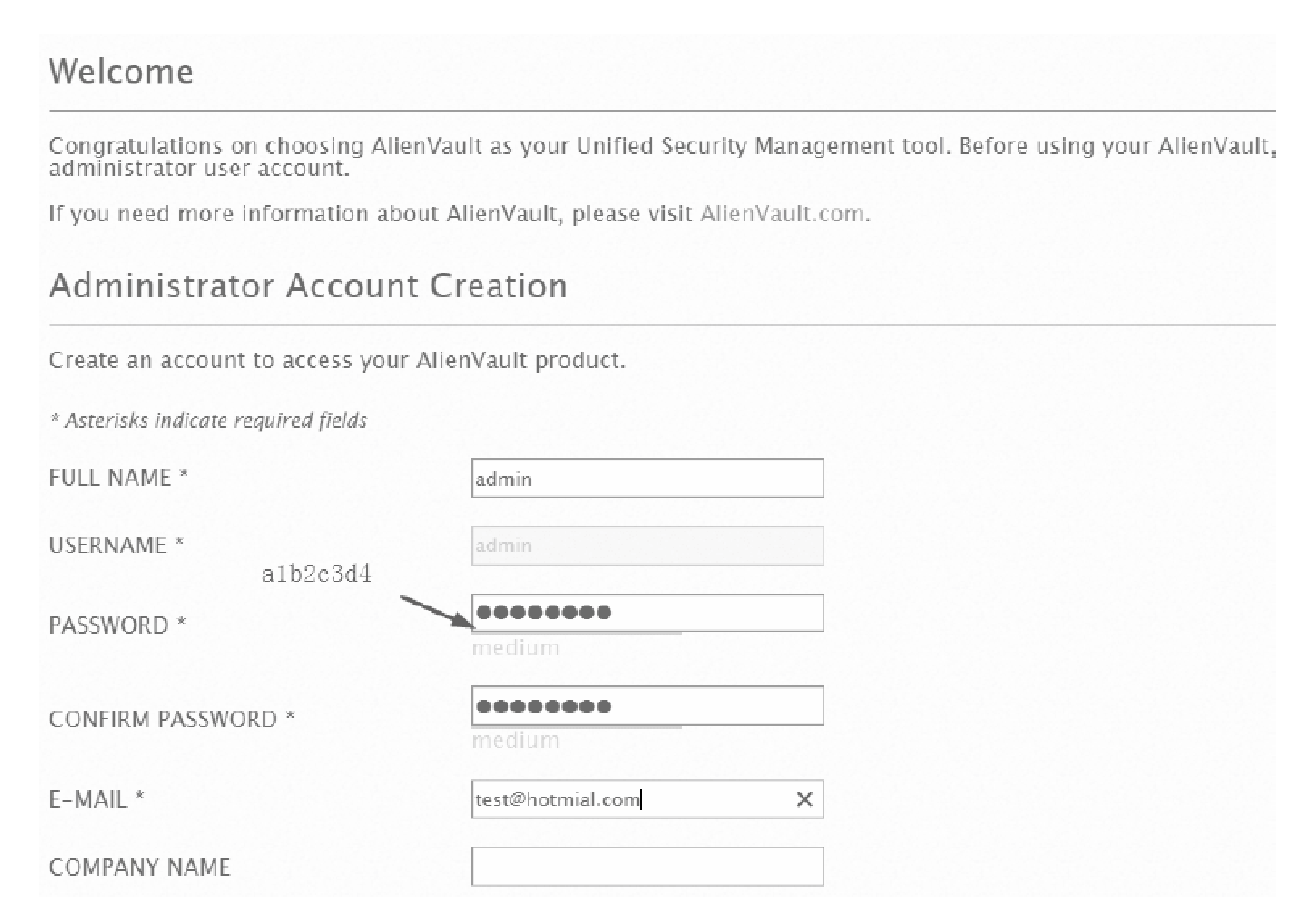


图 2-11　设置用户名和密码

对于实现原理，大家可参考脚本/usr/share/ossim-installer/auxscripts/support_info 的第 24~38 行。注意代码中 pwgen 是一个强大的密码生成工具，可以生成安全的强密码，并且支持设置排列方式随机生成密码。

2.4.3　遗忘 Web UI 登录密码的处理方法

通过终端控制台（输入 ossim-setup 进入）重置密码，路径为 0 System Preferences→4 Change Password→1 Reset UI Admin Password。

另一种方法是在命令下输入命令：

```
#ossim-reset-passwd admin
```

系统会产生一个随机的 8 位密码。

2.5　初始化系统

OSSIM Server 装完后，在浏览器地址栏中输入服务器 IP 地址，设定完管理员密码之后，即可开始系统配置。需要注意为了保证 Web 设置顺利，推荐用户在新装操作系统的浏览器下设置。OSSIM 初始化工作共分为 5 个步骤，图 2-12 所示为 Web 登录界面。

图 2-12　OSSIM 系统初始化登录

2.5.1　设置初始页面

系统经过 5 步配置，才能完成初始化，Web 代码位于/usr/share/ossim/www/wizard/steps/目录下，每步所对应的 PHP 代码如表 2-5 所示。

表 2-5　初始化配置 PHP 文件分布

序号	名称	功能描述	网页文件名称
1	NETWORK INTERFACES	配置网络接口（包括网络监控、日志采集和扫描）	step_1.php
2	ASSET DISCOVERY	资源发现	step_2.php
3	DEPLOY HIDS	HIDS 部署	step_3.php
4	LOG MANAGEMENT	日志管理	step_4.php
5	JOIN OTX	加入 OTX	step_5.php

大家掌握系统初始化功能和组成之后，接下来以单网卡混合式安装为例开始具体配置（也允许直接跳过配置向导）。

1. 配置网络

在配置网络过程中，首先遇到管理接口的配置界面，如图 2-13 所示。如果有多块网卡，可将管理口、嗅探口和日志收集口分别由 eth0、eth1 和 eth2 承担。如图 2-13 中虚线框中所示。

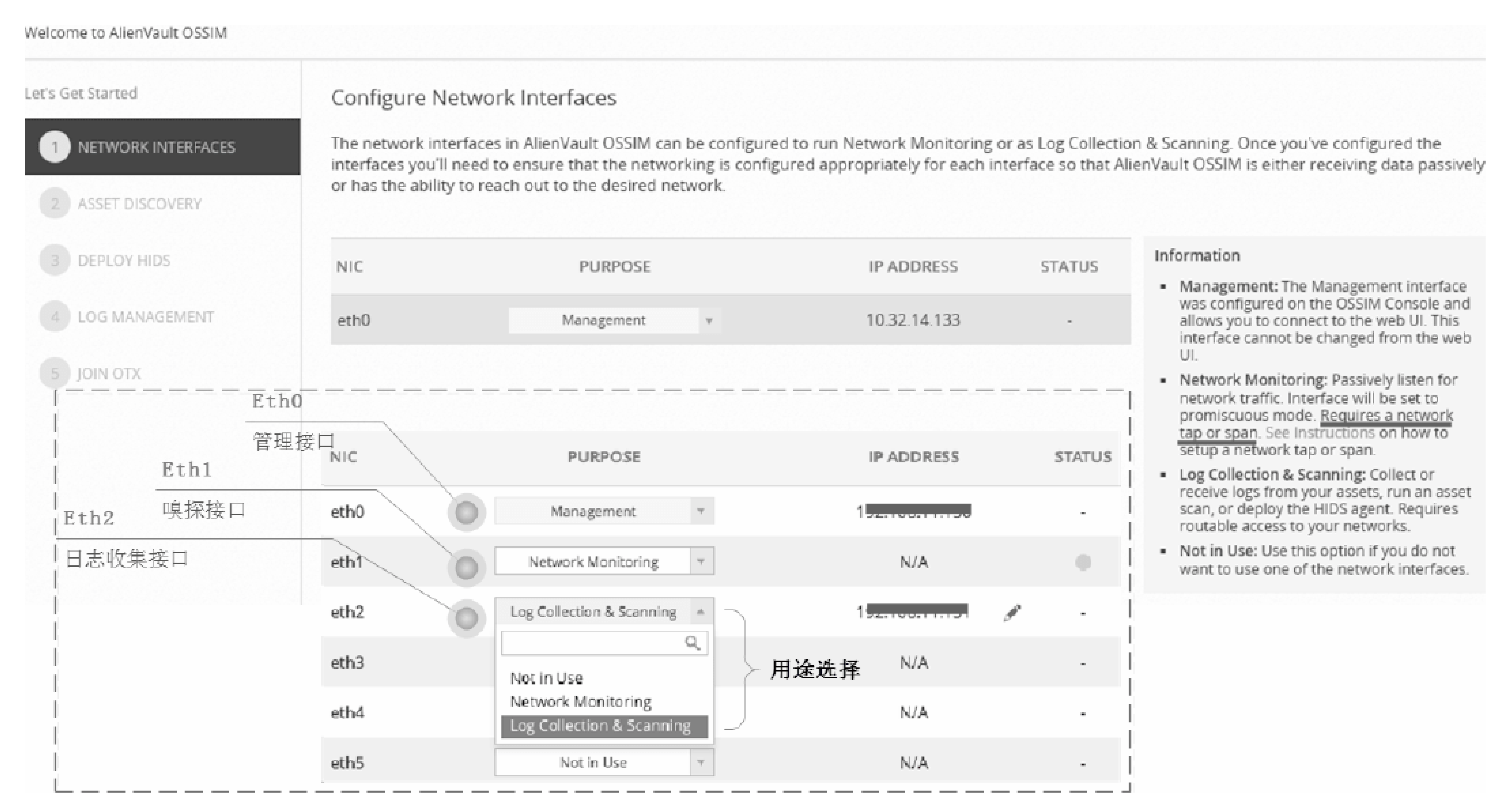


图 2-13　配置网络接口

这里强调网络监控接口必须采用 SPAN 或者网络 TAP 分流设备（Gigamon 解决方案），在选择这一步时，系统已经在后台开始使用 Nmap 工具扫描这台服务器所在的网段中的设备，目的是为下一步设置提前做好准备。

2. 发现网络资源

当单击下一步按钮后，系统开始设置所监控的网络环境中的各种资产（服务器、网络设备），并逐一添加到系统中，可以通过网络扫描发现，也可以通过导入 CSV 文件，还可以手工添加设备。实现操作界面如图 2-14 所示。

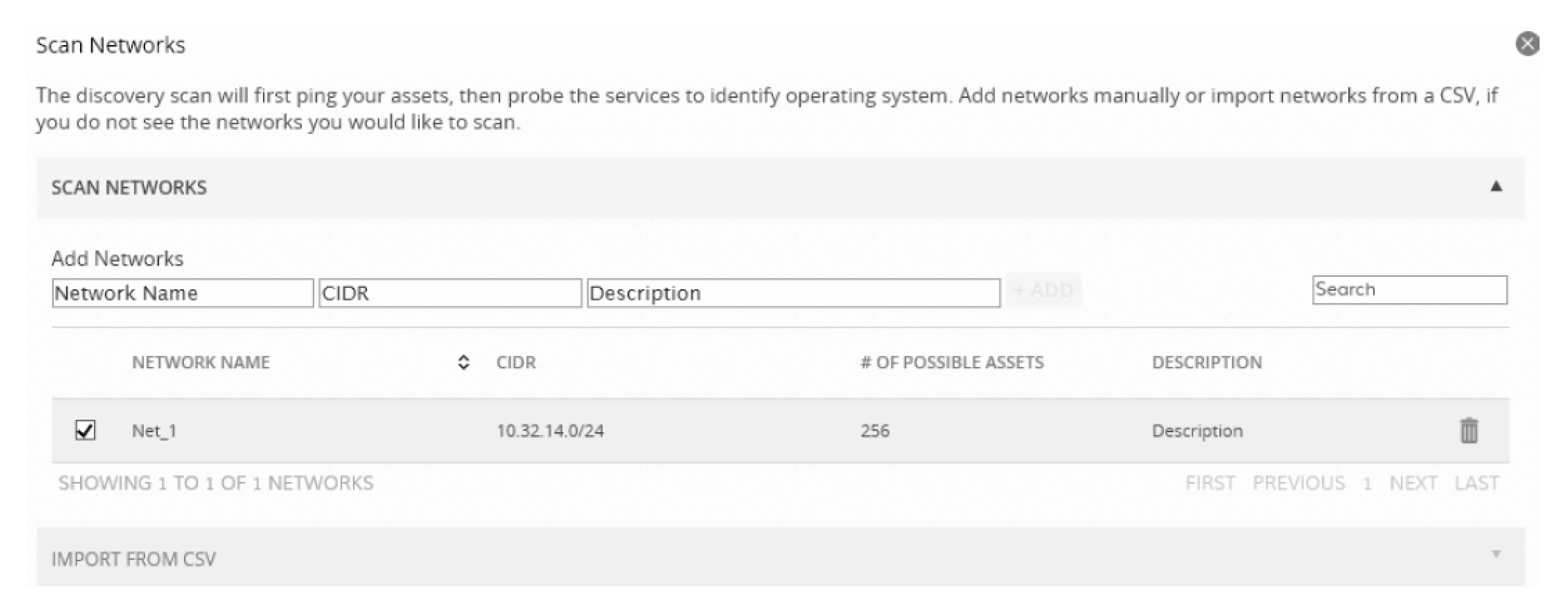


图 2-14　扫描监控网段

如果需要监控的网段比较多，系统还支持 CSV 文件导入方式，选择 IMPORT FROM CSV 选项栏，如图 2-15 所示。

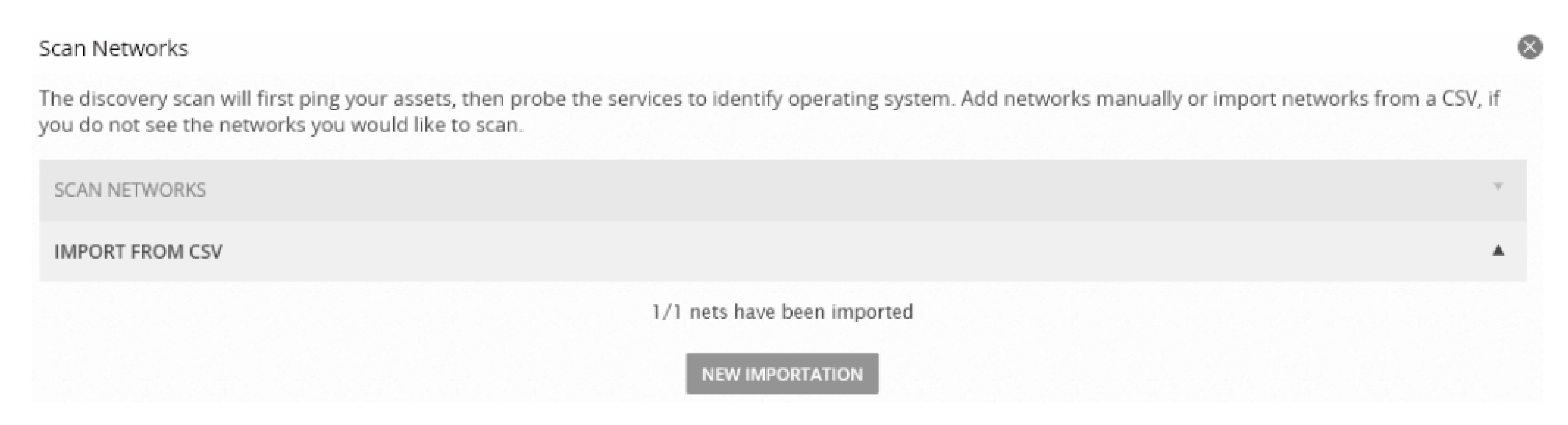


图 2-15　通过 CSV 导入

选择导入指定的 CSV 文件即可。下面看看这个文件的示例：

```
"Net_1";"10.32.14.0/24";"Description"
"Net_2";"10.32.15.0/24";"Description"
"Net_3";"10.32.16.0/24";"Description"
```

大家在编写这个文件时要注意保存为 CSV 格式。

这时，还需要设定扫描周期（每日、每周、每月执行），如图 2-16 所示。

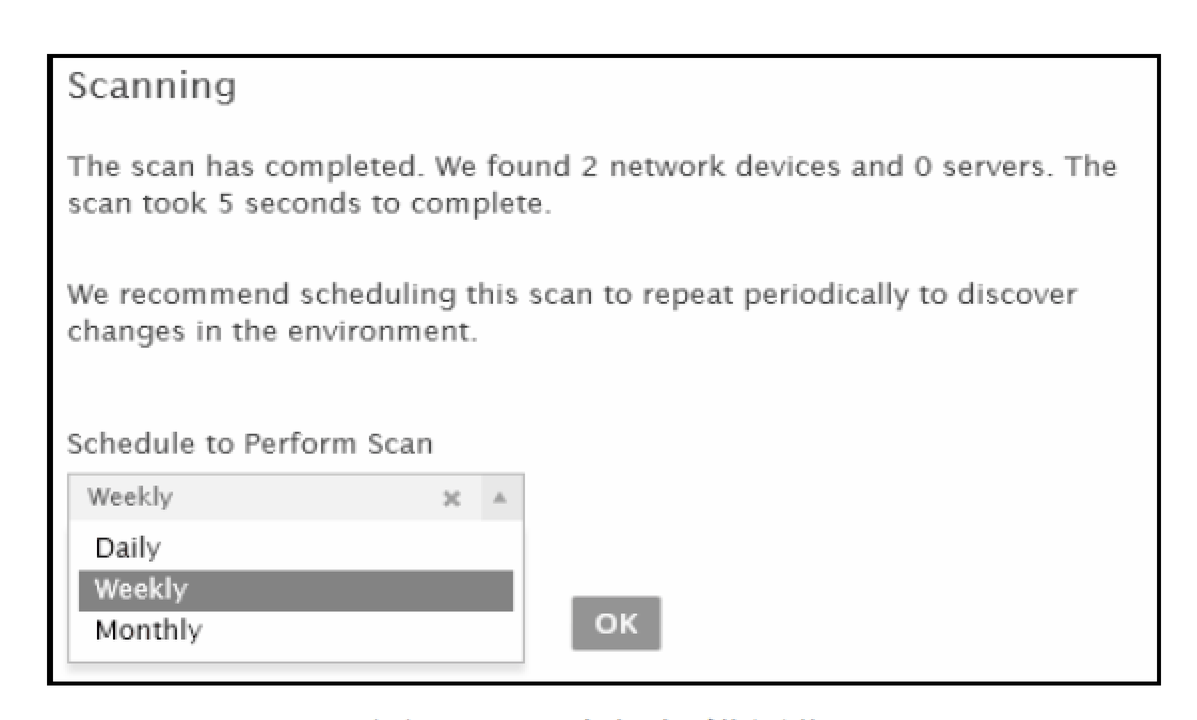


图 2-16　选择扫描周期

经过一段时间等待，扫描结果如图 2-17 所示。

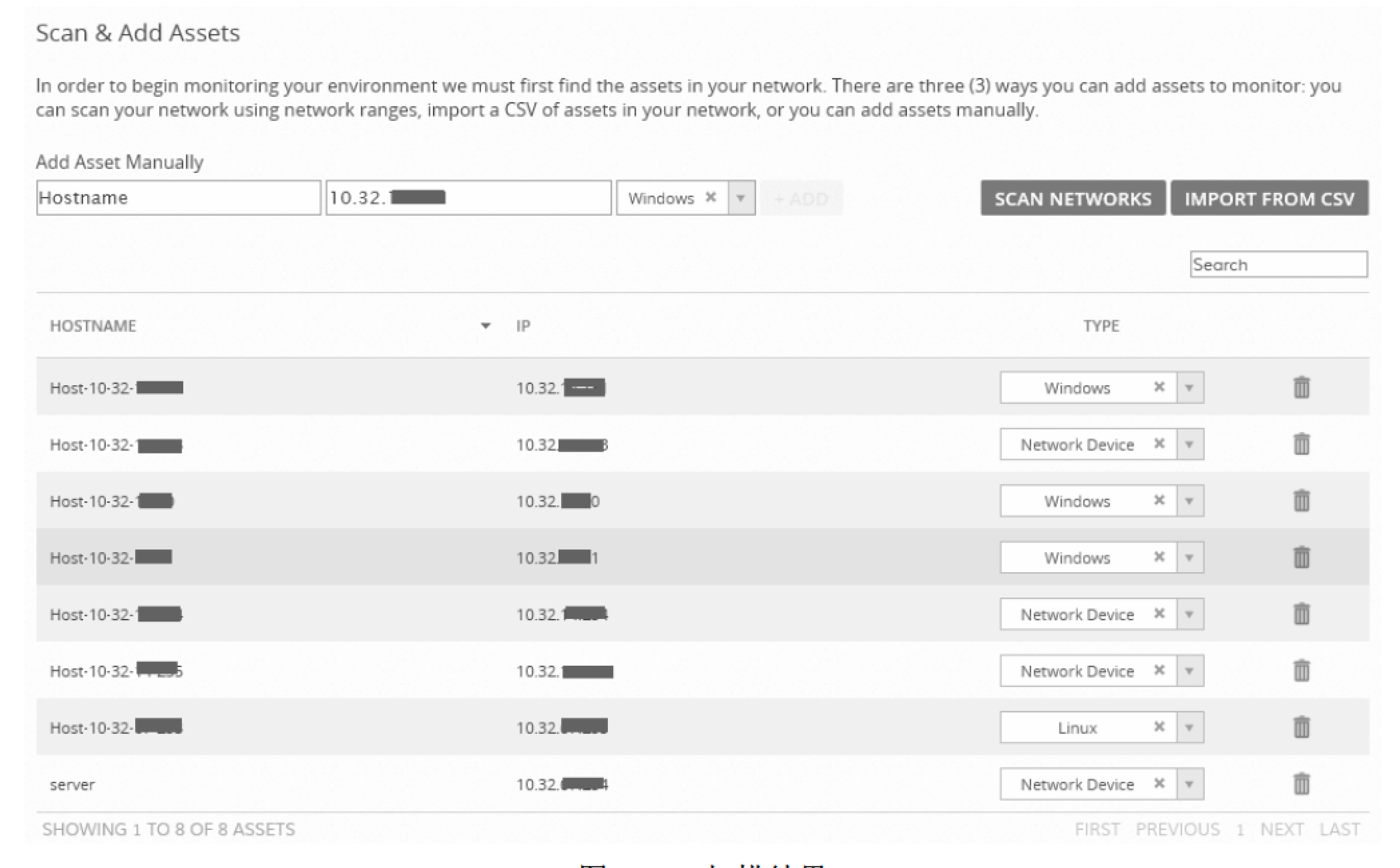


图 2-17　扫描结果

扫描结果比较准确，能够识别 Unix/Linux/Windows 系统以及常见网络设备，但和实际系统有些差别。另外，系统还支持手动导入 CSV 文件的方式来批量添加资产，通过选择 IMPORT FROM CSV 按钮实现，当然需要事先编写好 CSV 文件。如图 2-18 所示。

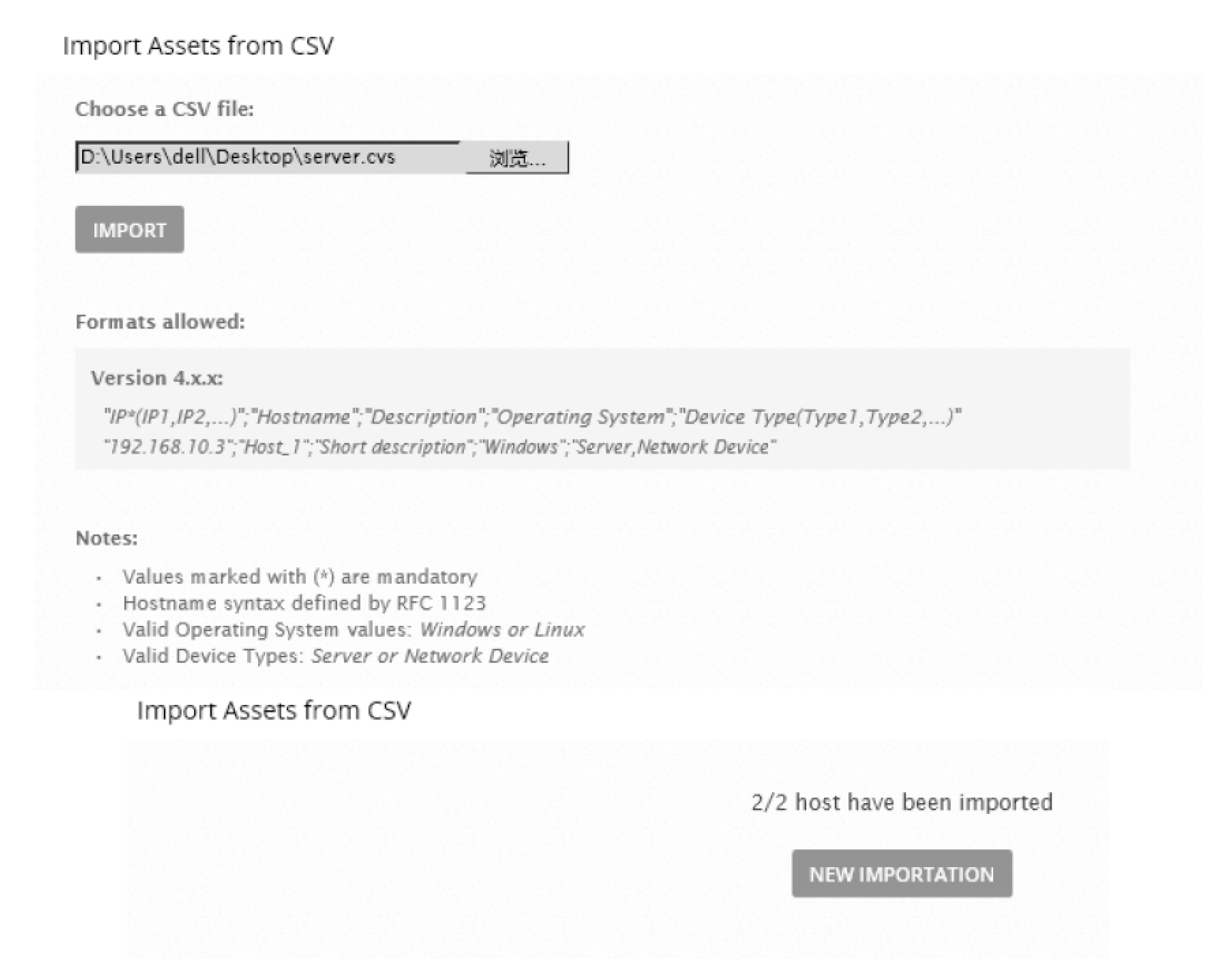


图 2-18　从 CSV 导入

大家尤其需要注意“Type”类型，这里尽量严格定义每个资产的类型，区分主机和网络设备，具体型号则越接近实体越好，这一步设置将关系到后续传感器状态的设置。

一个导入主机的 CVS 内容实例如下：

```
"10.31.14.131";"Server_1";"description";"Windows";"Server Device"
"10.31.14.132";"Server_2";"description";"Windows";"Server Device"
"10.31.14.133";"Server_3";"description";"Windows";"Server Device"
```

3. 部署 HIDS

这一步是为了部署 HIDS（基于主机的 IDS）而设置，主要目的是执行文件完整性监控，Rootkit 检测以及收集日志。注意：无论是 Windows 主机还是 Linux 主机部署 HIDS，先要关闭其防火墙，才能自动部署成功。

（1）Windows 系统

对于 Windows 系统而言，首先要在 UI 的左边栏部署主机列表中，选择若干台机器，输入域管理员用户名和密码，系统会将代理程序安装在所选择的主机之上。如图 2-19 所示。

Deploy HIDS to Servers

For these devices we recommend deploying HIDS in order to perform file integrity monitoring, rootkit detection and to collect event logs. For windows machines the HIDS agent will be installed locally, for Unix/Linux environments remote HIDS monitoring will be configured.

WINDOWS (4)
UNIX / LINUX (4)

Enter the domain admin account to install the HIDS agent. The username and password you provide will *not* be permanently stored, it will be used to deploy an agent to the selected assets.

Username
admin
Password
•••••••
Domain (Optional)

Deploy to the following hosts:
Net_1
Host-10-32-[redacted]
Host-10-32-[redacted]
Host-10-32-[redacted]
Others Hosts
Host-10-32-[redacted]

DEPLOY

图 2-19　将 HIDS 部署到主机

在 Windows 系统中成功自动部署 HIDS 如图 2-20 所示。

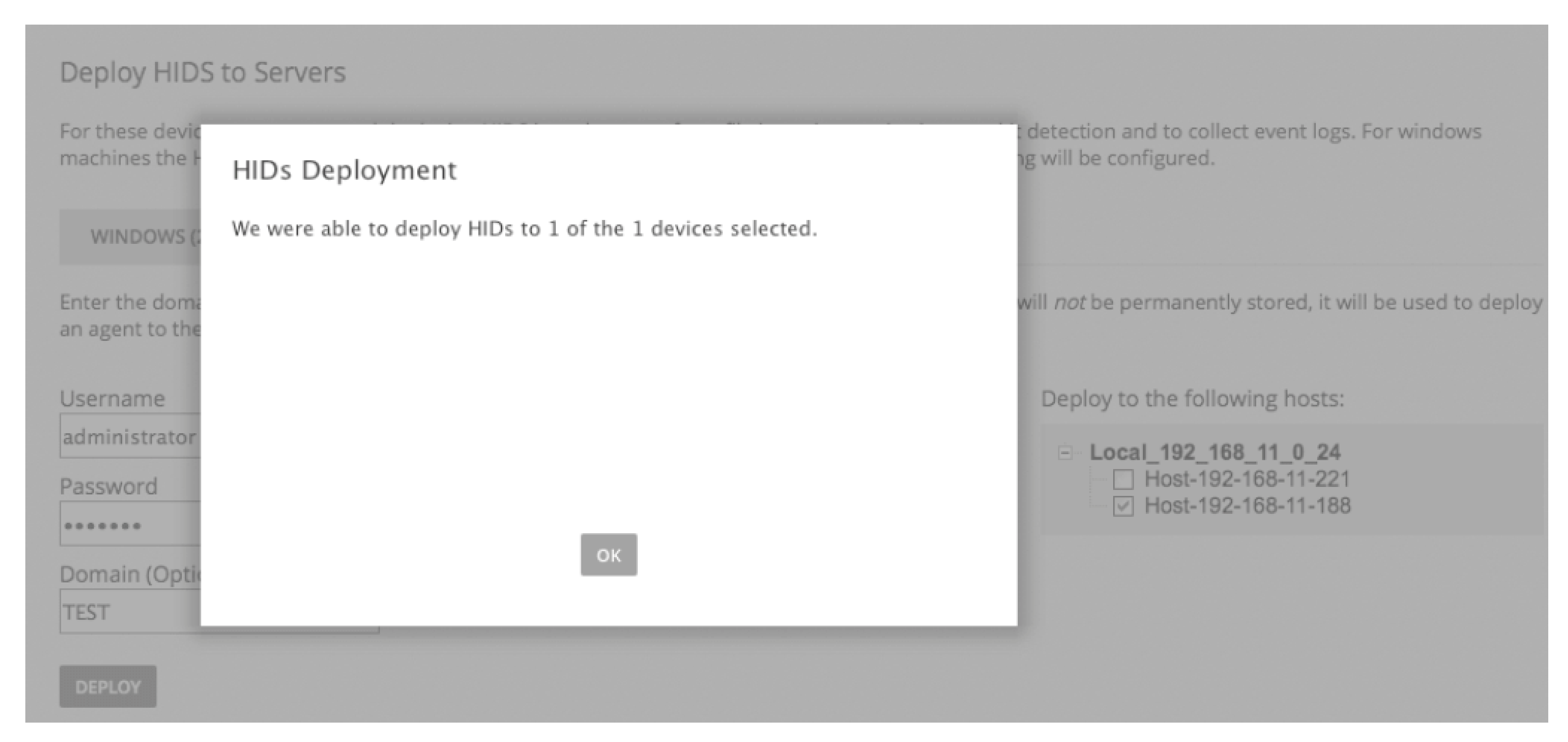


图 2-20　一台主机被成功部署

此时，已由系统向导将 HIDS Agent 安装到远程 Windows 客户端，安装默认路径为 C:\Program Files\ossec-agent\。通过运行该目录下 win32ui.exe 程序，可以查看代理工作状态。使用该向导适合批量部署代理，在本书第 9 章，将详细介绍手动 OSSEC Agent 安装方法。

（2）Linux 系统

对于 Linux 系统添加时，选择主机并输入 root 用户和密码，接着系统开始安装代理，该密码出于主机监控目的，系统会保存下来，以便定期访问选定的资产。注意也有不少 Unix/Linux 不支持自动安装，那么读者就需要手动安装 HIDS。在 Linux 系统中成功自动部署 HIDS 如图 2-21 所示。

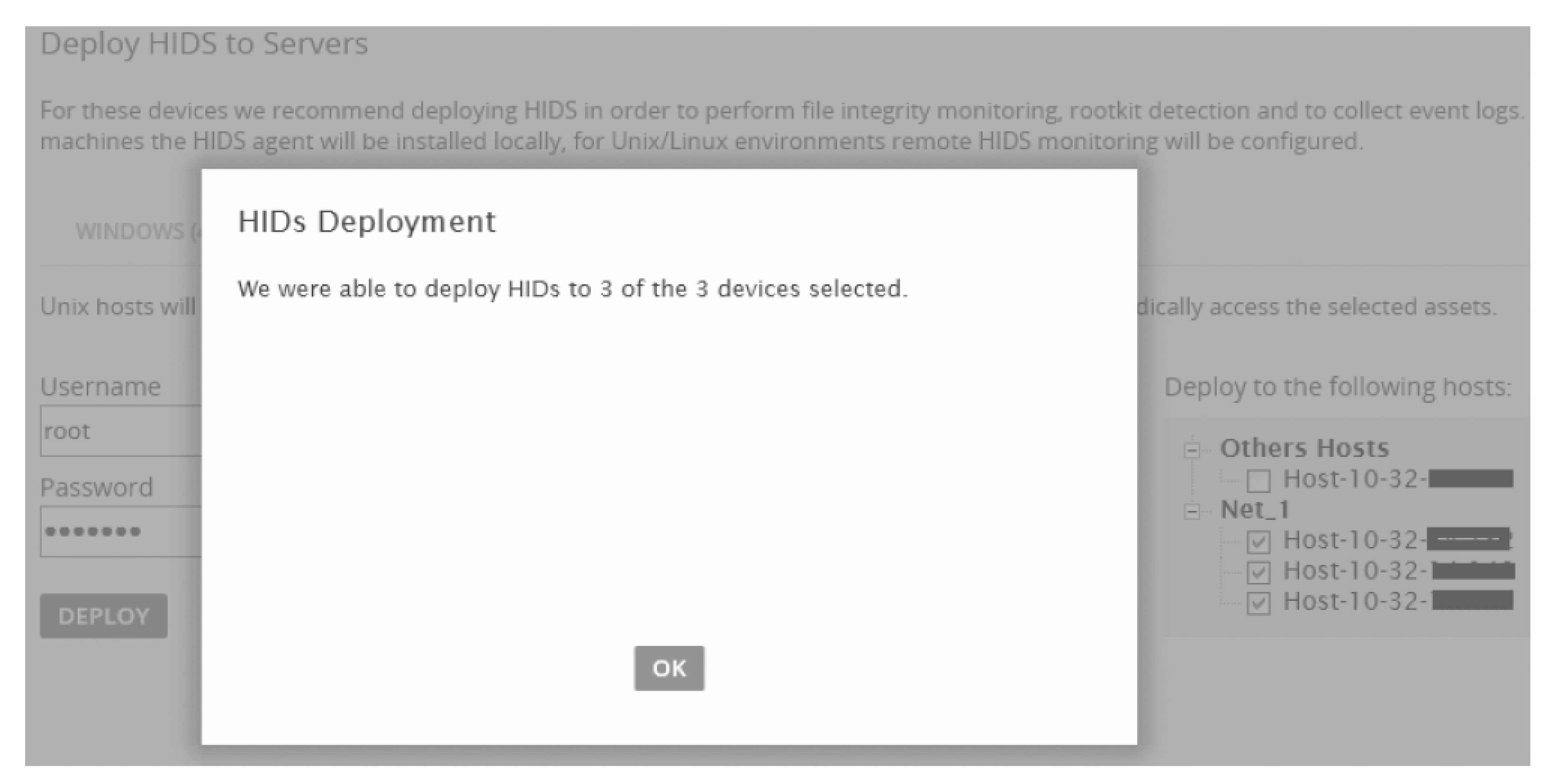


图 2-21　3 台主机被成功部署

故障处理：在第 3 步的部署当中会遇到系统报错，出现红色字体“There are no servers on your network. Return to the asset discovery page by clicking back to scan your network or by adding servers manually. ”解决方法是，退回到第 2 步“Asset Discovery”，选择“Scan Networks”按钮，在弹出对话框中选择当前监控网段，例如“10.32.14.0/24”，最后单击“Scan Now”按钮，这时再继续操作就没有报错提示。

4. 日志管理

这一步对资产发现扫描，探测到了 5 个网络设备并进行设置，应尽量调查设备厂家、型号、版本信息，这样系统可以对相应设备启用数据源插件。如图 2-22 所示。如没有合适的，就选择最接近的一项。

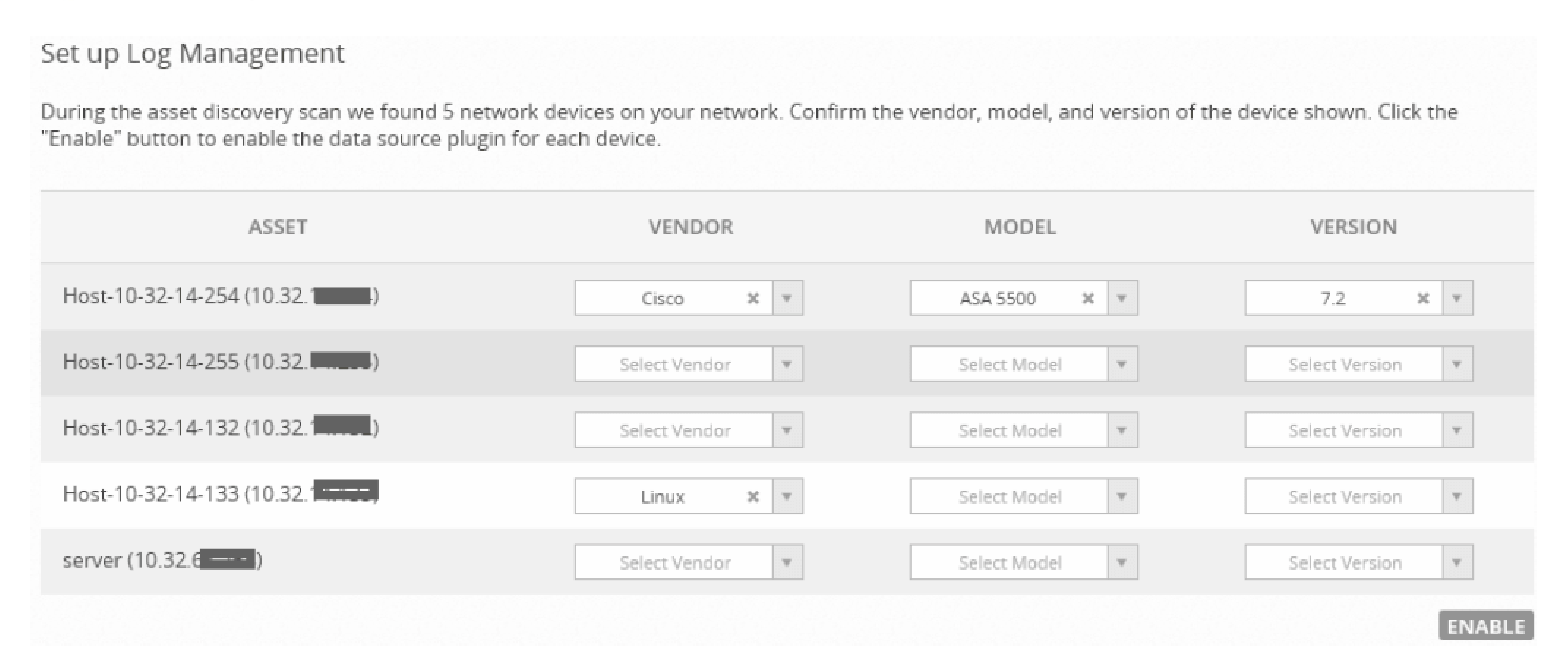


图 2-22　设置日志管理

当插件配置正确之后，会向 OSSIM 主机发送日志，当 OSSIM 主机接收到日志后“接收

数据”的指示灯会变成绿色，这时可以单击完成按钮（至少从一台设备成功接收到日志数据，按钮才可单击，否则只能跳过该项）。

如图 2-23 所示，已为 Cisco 192.168.11.100 这台设备启用插件，并成功接收到这台设备的日志数据。

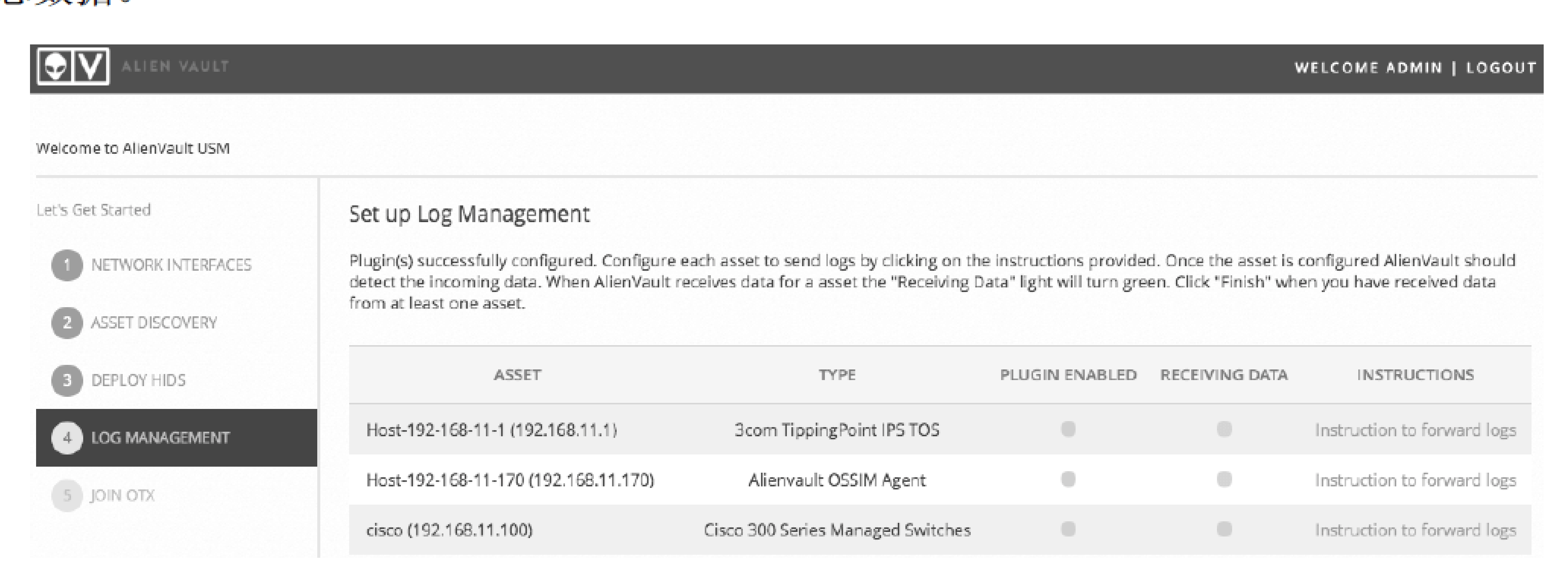


图 2-23　检测资产的插件状态

5. 将 OSSIM 加入 OTX

设置 OTX 时，首先单击申请账户并登录 www.alienvault.com 网站，注册成功后用户会收到 64 位令牌（Token），将这串数字保留下来，在 OSSIM Web UI 的 Configuration→Administration→Main →Open Threat Exchange 下输入令牌并激活，注意此账户一定要激活才生效。如图 2-24 所示。

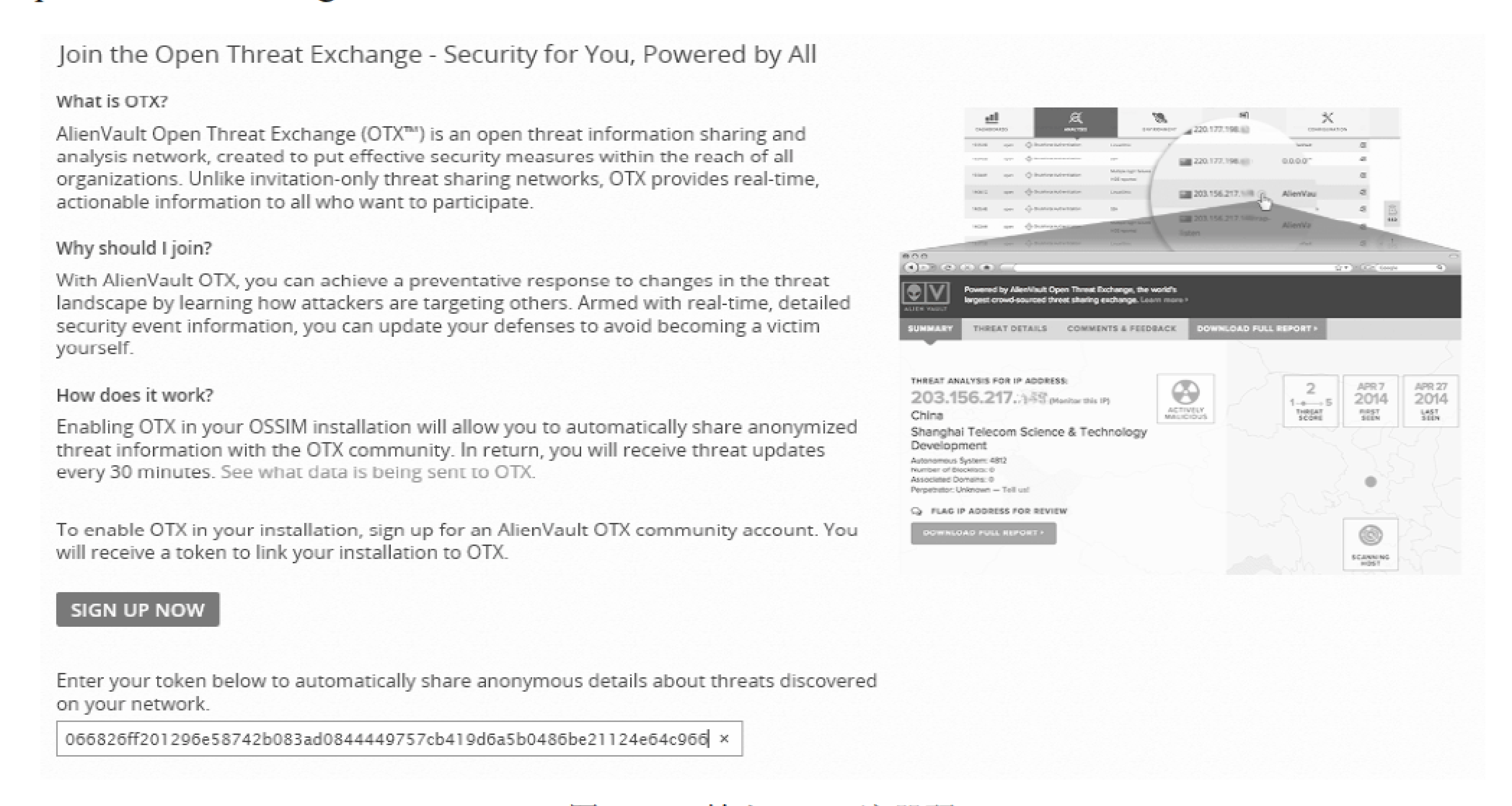


图 2-24　输入 OTX 注册码

最后配置完成选择“EXPLORE ALIENVAULT USM”按钮，完成整个配置向导，如果有多个 VLAN，那么需要继续添加 Sensor 的信息，这时需要选择“CONFIGURE MORE DATA SOURCES”，如图 2-25 所示。

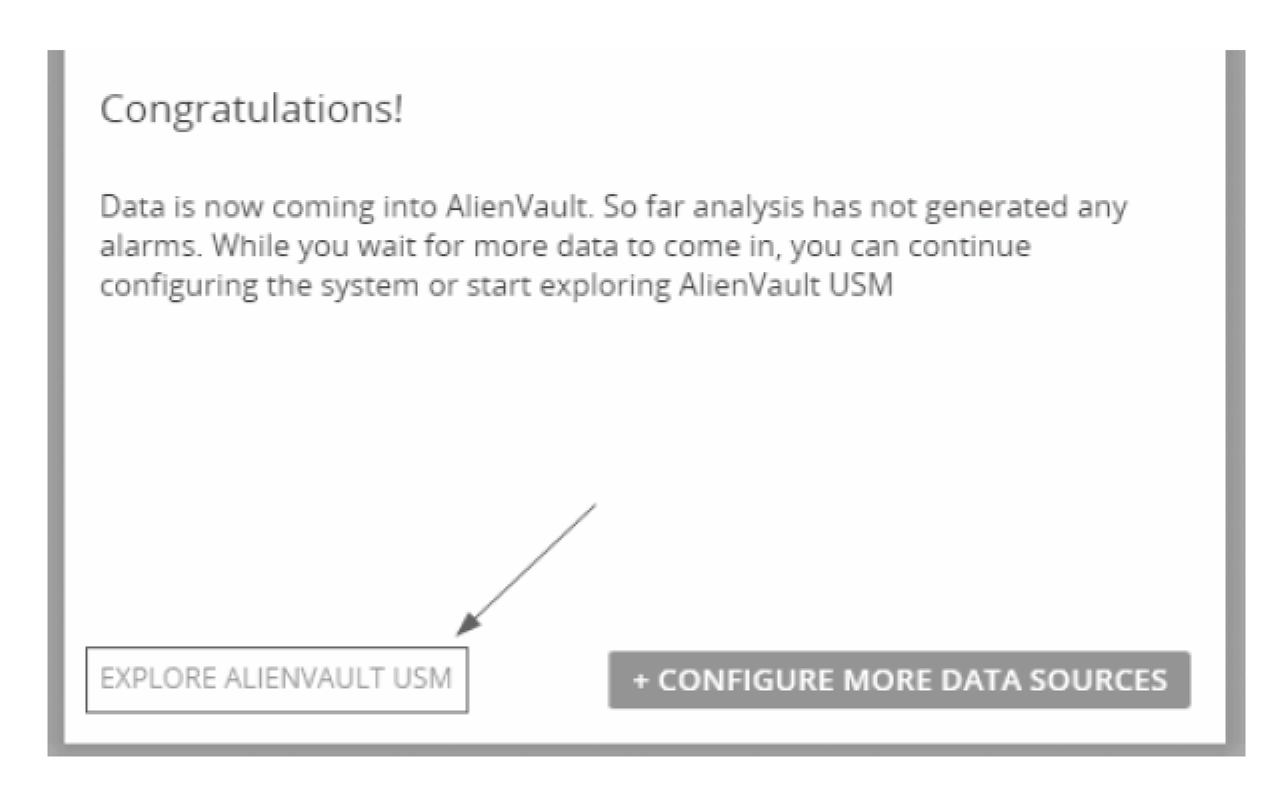


图 2-25　配置完成

2.5.2　OTX——情报交换系统

随着对攻击方式的研究不断深入，攻击者产生恶意代码的成本不断降低，而很多企业的防护产品依然依赖于传统的样本采集和特征码分析的分发机制。恶意代码样本数量呈现雪崩式增长，直接导致传统的依赖特征库的现有防护体系不堪重负。无论是特征库容量还是威胁响应速度，都难以跟上恶意代码的增长。为了解决这一问题，Alienvault 对物联网资源（域名、IP、URL 等）进行威胁分析和 IP 信誉评级，并在 OSSIM 的 SIEM 内事件收集加入了针对目标资源安全信誉服务查询，这就是 OTX。OTX 全称是 AlienVault Open Threat Exchange（简称 AV-OTX，Alienvault 公开威胁交换项目），它是建立在 USM（统一安全管理平台）之上的系统，依靠 OTX 便可获得相应的安全信誉评级信息，进而根据结果调整防护策略。

根据威胁来源不同，可以将威胁分为内部威胁和外部威胁两种。内部威胁是指系统的合法用户以故意或非法方式进行操作所产生的威胁，除此之外，还有来自 Internet 庞大系统的外部威胁，外部威胁情报通过从你的整个网络搜集而来的本地威胁情报进行增强，并与环境数据关联。

这些实时威胁数据，来自于安全情报交流社区。在这个社区中有全世界超过 170 多个国家，30000 多个用户成功部署 OSSIM，各社区上报威胁数据，其目的是更全面、更多样化地防范各种攻击模式，就好比像全球发布的“通缉令”一般。

OTX 的显示特性与 Norse（可以反映全球黑客网络攻击实时监控数据，http://map.ipviking.com/）显示的数据有些类似，所不同的是，OTX 主要展示的是一种威胁交换分享，图 2-26 圆圈中的动态变化的数字，不但反映出网络威胁出自那些国家地区，能反映出动态变化过程，它的主要记录位于/etc/ossim/server/reputation.data 文件中，其中包含了已知恶意 IP 地址数据库检查的 IP 信誉度评价。

那为什么要对 IP 进行信誉评级呢？传统安全解决方案采用的是判断行为“好”或“坏”，然后再进行“允许”或“拦截”之类的策略，不过随着高级攻击日益增多，这种分类方法就不足以应对这种威胁。许多攻击在开始时伪装成合法的流量进入网络，得逞之后再实施破坏，由于攻击者的目标是先混进系统，所以需要对其行为进行跟踪，并对其行为进行信誉评级，以确定是否合法。那么 IP 信誉评级的依据是什么？在 AlienVault 中的 IP 信誉系

统中，利用海量资源（客户反馈、网络爬虫、合作伙伴、安全扫描、蜜罐网络等）从不同信息源获取的事件分析结果进行关联分析得到，作为用户仅仅通过 OTX 接口就能进行查询，即可获得相应的信誉评级，通过多源、多维度信息综合分析，从而从单一层面来解决新型威胁的技术难题。

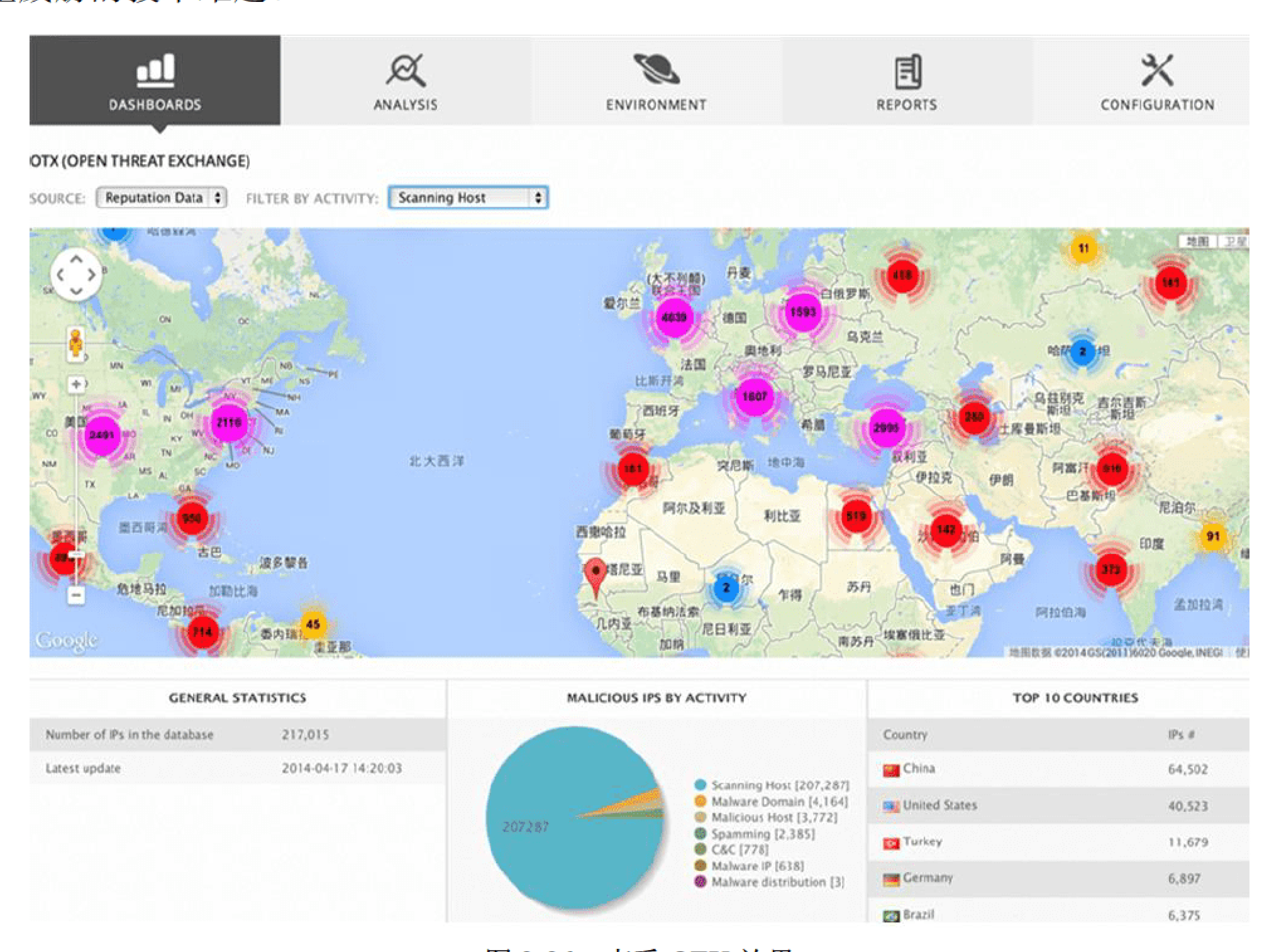


图 2-26　查看 OTX 效果

OSSIM 中反映出的 IP 特征与 IP 地址相关，包括 IP 地址的域名、地理位置以及操作系统和提供的服务功能等。通过这些信息构建出全球 IP 信誉系统。如果没有使用 OSSIM 系统，也可以到 http://www.commtouch.com/check-ip-reputation/查询。例如，查询结果显示：This IP address has not been used for sending Spam（这代表此 IP 地址没有被用来发送垃圾邮件）。全球许多组织维护了一些黑名单，用来跟踪记录 IP 的信誉度，目前还有个反滥用项目 Web 站点有 www.anti-abuse.org/multi-rbl-check、www.kloth.net/services/dnsbl.php 等，通过查询结果，可以很详细地了解你的服务器 IP 在黑名单的情况。如果有 IP 出现在多个黑名单上则可以确定是有问题 IP。一旦找到了有嫌疑的 IP 地址就需要确定以下四个问题：

- IP 地址的地理位置位于何处？
- 什么组织机构负责该 IP 地址？
- 该地址的不良信誉如何？
- 哪些 DNS 条目指向了该 IP 地址？

而这 4 个问题会在 OTX 中得以解决，下面看看如何激活 OTX 功能。如图 2-27 所示。

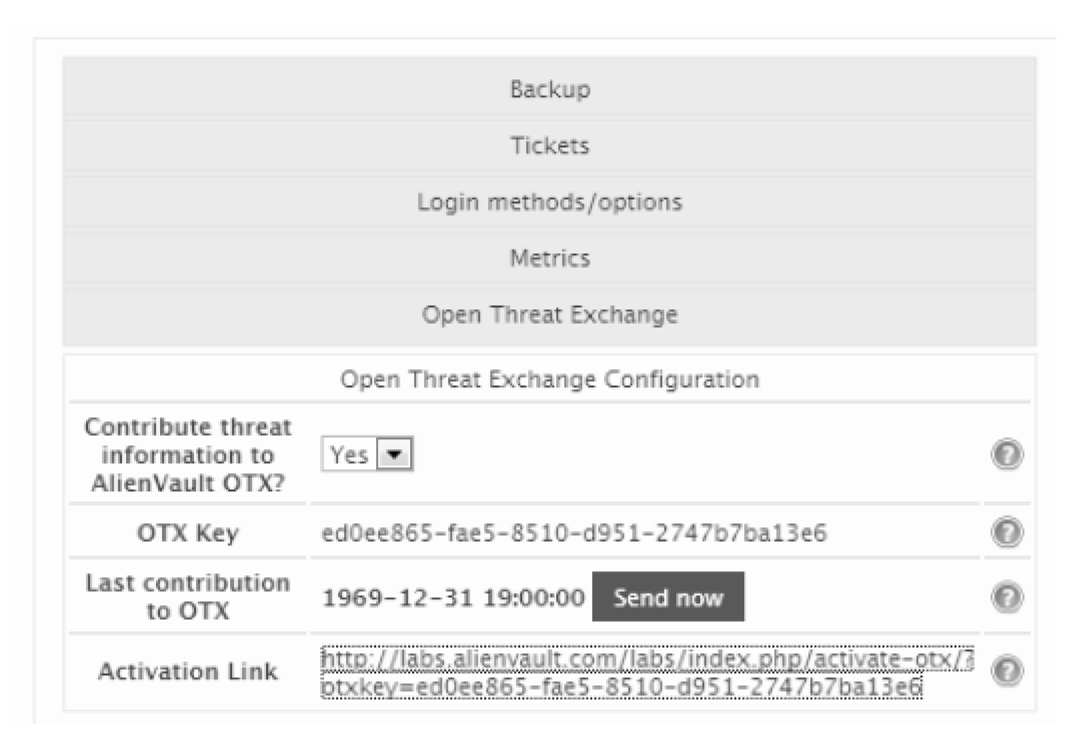
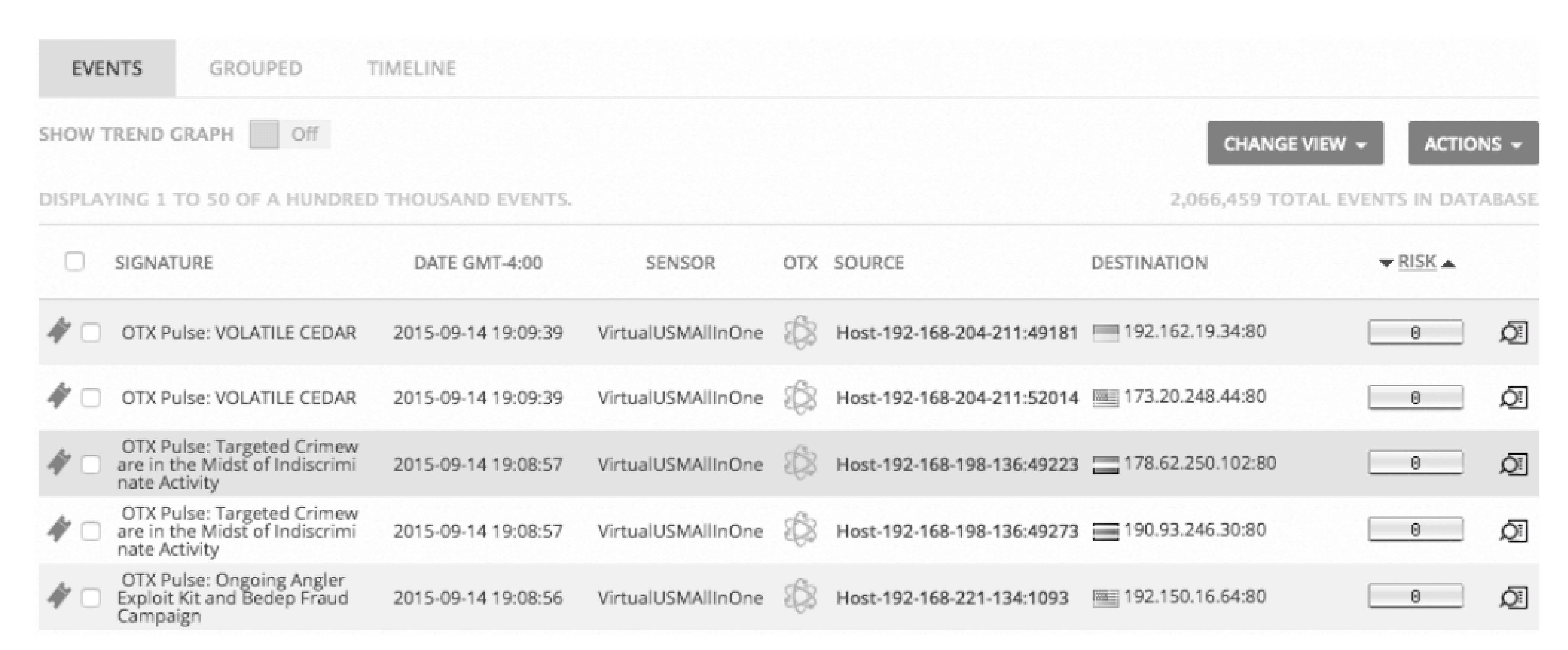


图 2-27　查看 OTX 激活码

当你的 OSSIM 系统加入 OTX，OTX 功能才能被启用，再次查看 SIEM 事件，其后多出了 OTX 项，并同时出现了一个黄色的图标，其中的 OTX 交换数据每隔半小时更新一次。如图 2-28 所示。



SIGNATURE	DATE GMT-4:00	SENSOR	OTX	SOURCE	DESTINATION	RISK
OTX Pulse: VOLATILE CEDAR	2015-09-14 19:09:39	VirtualUSMAllInOne		Host-192-168-204-211:49181	192.162.19.34:80	0
OTX Pulse: VOLATILE CEDAR	2015-09-14 19:09:39	VirtualUSMAllInOne		Host-192-168-204-211:52014	173.20.248.44:80	0
OTX Pulse: Targeted Crimeware are in the Midst of Indiscriminate Activity	2015-09-14 19:08:57	VirtualUSMAllInOne		Host-192-168-198-136:49223	178.62.250.102:80	0
OTX Pulse: Targeted Crimeware are in the Midst of Indiscriminate Activity	2015-09-14 19:08:57	VirtualUSMAllInOne		Host-192-168-198-136:49273	190.93.246.30:80	0
OTX Pulse: Ongoing Angler Exploit Kit and Bedep Fraud Campaign	2015-09-14 19:08:56	VirtualUSMAllInOne		Host-192-168-221-134:1093	192.150.16.64:80	0

图 2-28　OTX 效果

如果希望在地图上显示这些 IP 地址，就可以使用 OTX 功能，位置在 DashBoards→OTX 菜单中，如图 2-29 所示。

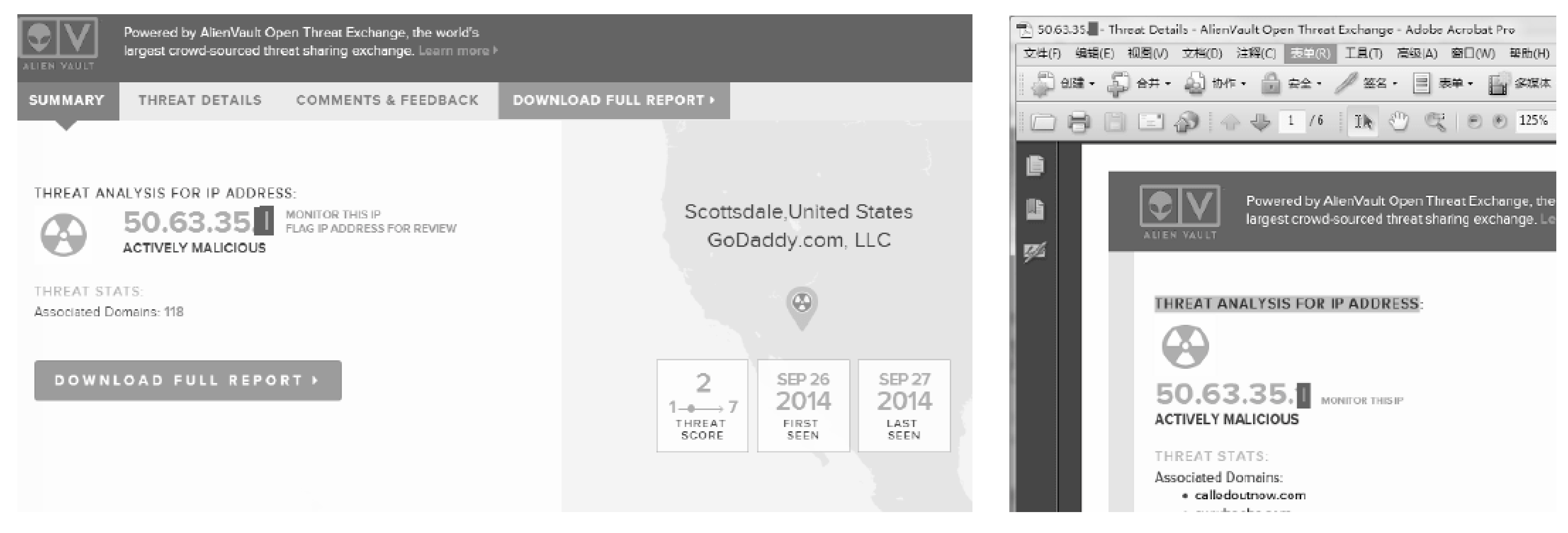


图 2-29　通过 OTX 查询可疑 IP 并获得详细 PDF 格式报告

2.6 VMware ESXi 下安装 OSSIM 注意事项

虚拟化能使服务器的资源被更有效地利用，通过虚拟化能将一台服务器的资源利用率最大化。在 OSSIM 部署时采用虚拟化技术，可以方便地制作系统克隆，通过快照技术能恢复到以前制定的状态下。下面介绍在 ESX i5 中部署 OSSIM 系统时，需要对网卡和硬盘的特殊配置要求。

2.6.1 设置方法

对于 VMware ESXi 安装软件和文档可在 https://www.vmware.com/cn/ 中找到，具体方法本书不再详细介绍。有 3 个特殊的地方还要强调:

（1）网卡的配置，在 OSSIM 服务器上需要安装双网卡，选择适配器型号为 e1000，若选择其他网卡，那么在 OSSIM 安装过程中无法识别网卡，磁盘可用空间推荐 1TB 及以上。

（2）ESX 的虚拟主机通过 vSwitch 来连接网络，vSwitch 是通过主机上的物理网卡作为上行链路与外界网络进行连接，这里 Virtual Switch（vSwitch）相当于一个虚拟的二层交换机，在虚拟机中安装 OSSIM USM 时，建议安装两块网卡，一块用于网络管理，一块用于 SPAN。如图 2-30 所示。

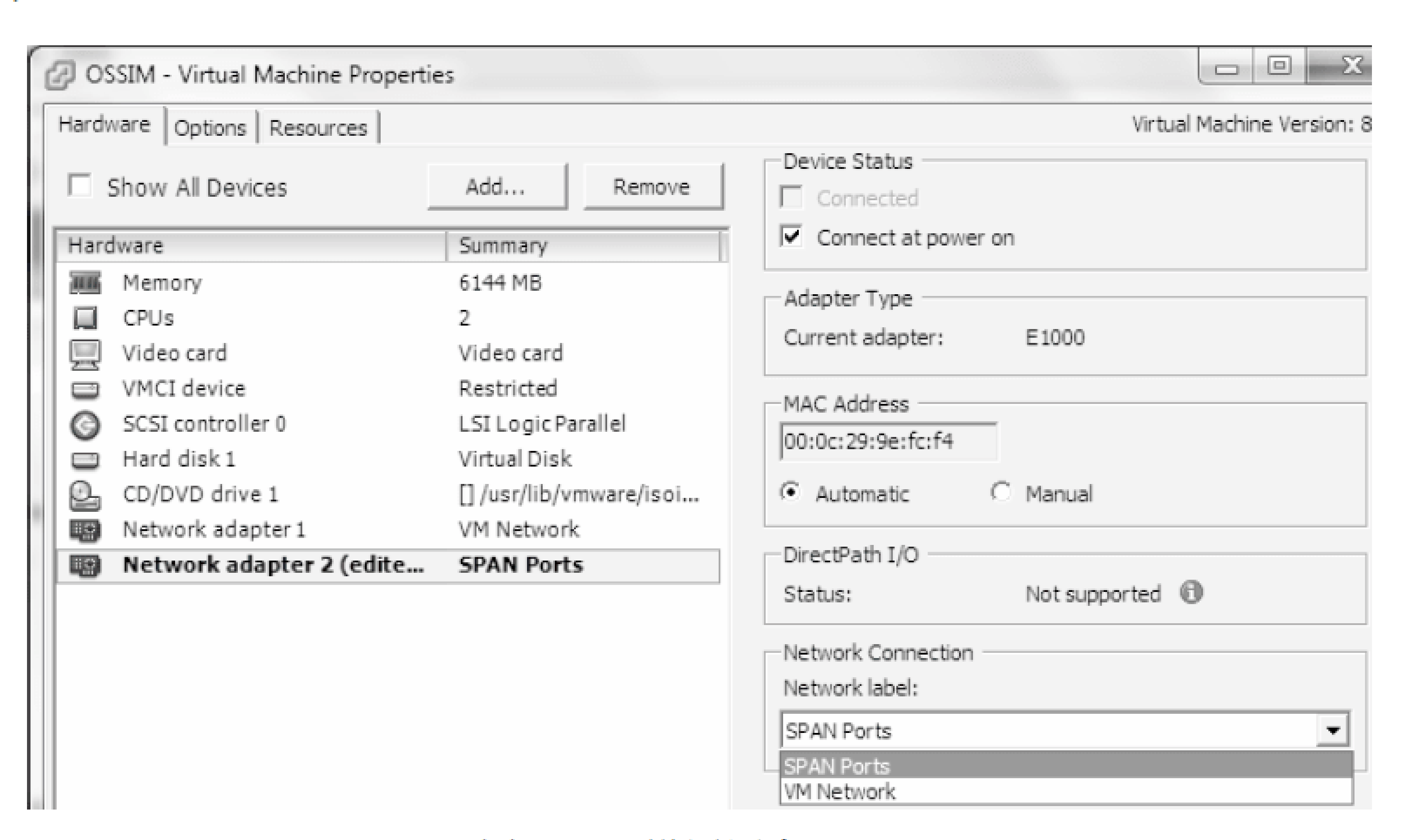


图 2-30　设置网卡 SPAN

（3）我们需要在 vSwitch 的安全控制选项中，设置 Promiscuous Mode 虚拟交换机的混杂模式。打开该模式，所有虚拟网卡的报文就会复制到 vSwitch 的所有 vPort 上面。我们就可以通过 OSSIM 实现监控。如图 2-31 所示。接着我们需要在 OSSIM 控制台上进入 Sensor 设置，将 eth1 设置为 SPAN 口的网卡，如图 2-32 所示。

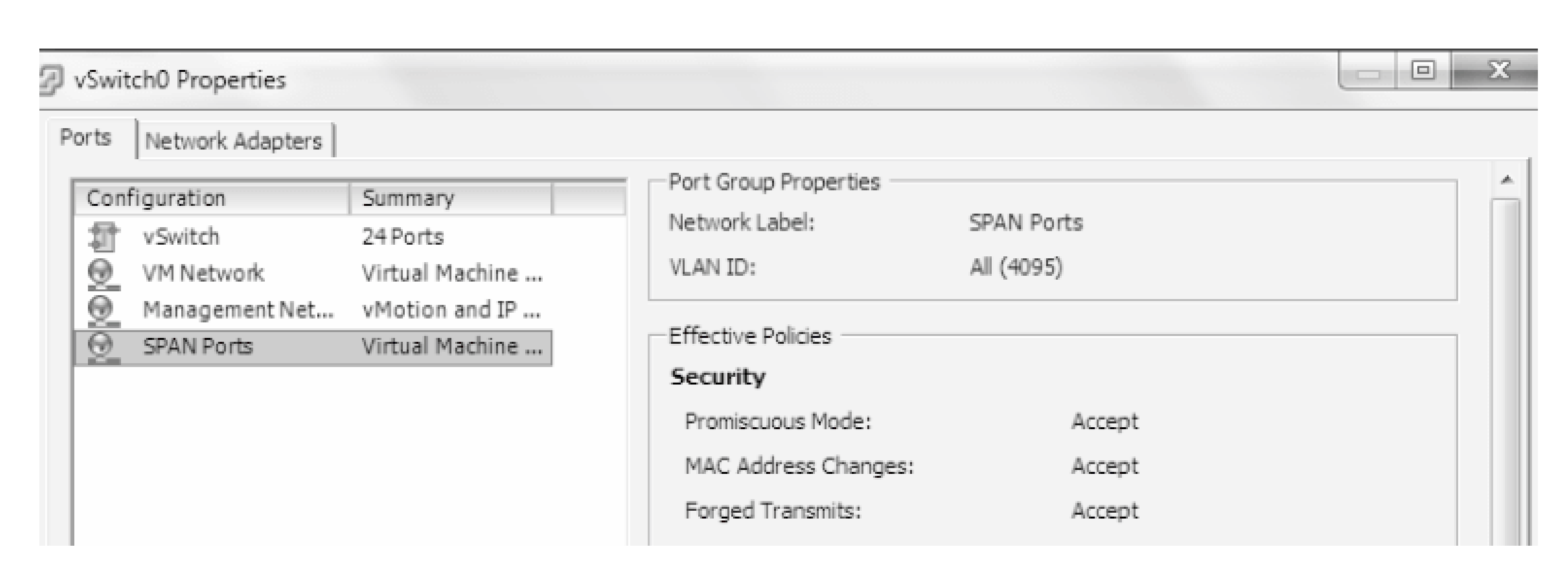


图 2-31　查看 SPAN 设置

更多 ESXi 设置大家参考下面链接：

```
    http://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.vsphere.net
working. doc_50%2FGUID-8D1768B0-074D-4F06-9931-2BE4777D35F8.html
```

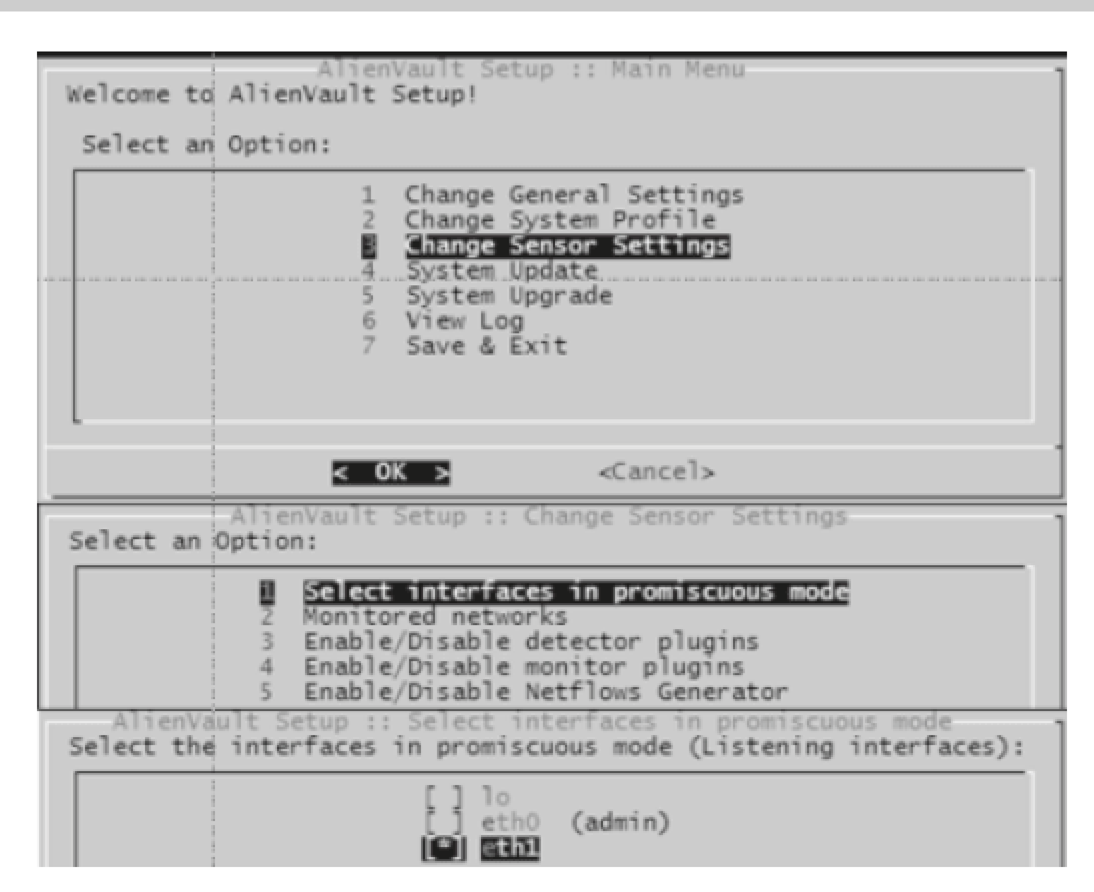


图 2-32　OSSIM 控制台下设置网卡

2.6.2　虚拟机下无法找到磁盘的对策

在虚拟机下安装 OSSIM 系统时会出现无法找到硬盘的情况，解决方法是，进入添加设备界面，选择“Add Device”，然后进入设备驱动选择界面，选择“BusLogic MutiMaster SCSI（BusLogic）”，依次单击“OK”按钮，选择“Done”按钮完成操作。如果在 ESXi 下创建虚拟机时，SCSI 控制器建议选择成“LSI Logic Parallel”。

为什么会出现这样的问题？其实出现问题的原因就在于 VMware 默认将硬盘设置为“SCSI-BusLogic”模式，所以在安装一些 Linux 发行版时，在系统启动过程中内核没有加载相应驱动程序，故无法找到磁盘。

2.7　OSSIM 分布式安装实践

首先要谈谈 OSSIM 传输安全的问题，在分布式系统中 Sensor 收集的日志要安全传输到

OSSIM Server 必须有传输加密机制，以确保收集到日志的安全性。在 OSSIM Server 中集成有权威的、可信赖的第三方认证授权中心，称为 CA 认证中心，它运用对称和非对称加密算法、数字证书、数字签名等技术建立起一套严密的身份认证系统。

2.7.1 基于 OpenSSL 的安全认证中心

SSL 为 Sensor 和 OSSIM Server 之间的通信实体，提供一个安全通道，保证数据的机密性。在服务器端进行强制认证，在客户端进行可选的认证服务，从而防止监听、篡改、消息伪造，OSSIM 上配置的 Apache 服务器支持 SSL 协议，当客户端浏览器试图连接一个具有 SSL 认证功能的 Apache 服务器时，会唤醒 SSL 会话，浏览器才能成功完成认证，用户能确认其浏览器连接到正确的服务器，而不是连接到一些想盗取用户密码等重要信息的虚假的服务器上。

2.7.2 安装步骤

（1）首先是在监控端（Sensor 所在 VLAN）安装好 OSSIM 配置插件文件，例如 Snort，ID 号为 1001，它需要在 snort.conf 中指定日志输出类型，这个类型可以分为 syslog、tcpdumplog、alert、unfield 几类，OSSIM 默认的为 unfield 输出。

（2）配置代理的配置文件（config.cfg），指定 outserver IP 为发送服务器端的 IP 地址，添加相应插件的文件路径，例如 snort=/etc/ossim/agent/plugins/snortunified.cfg，该 cfg 文件中可以指定获取到的日志信息路径，并匹配正则表达式。

（3）在 Web 页面添加 Sensor。分布式 OSSIM 系统配置的关键在各个 cfg 配置文件上，只要指定相应的日志文件路径，那么 Sensor 就能够分析插件所捕获的信息，并发送到 Server 端。

（4）一般在 OSSIM Server 上安装双网卡，一块用于接收 SPAN 的流量，另一块用于远程管理，从而避免在大负载情况下单网卡负担过重。比如交换机与 OSSIM 的连接方式如图 2-33 所示。

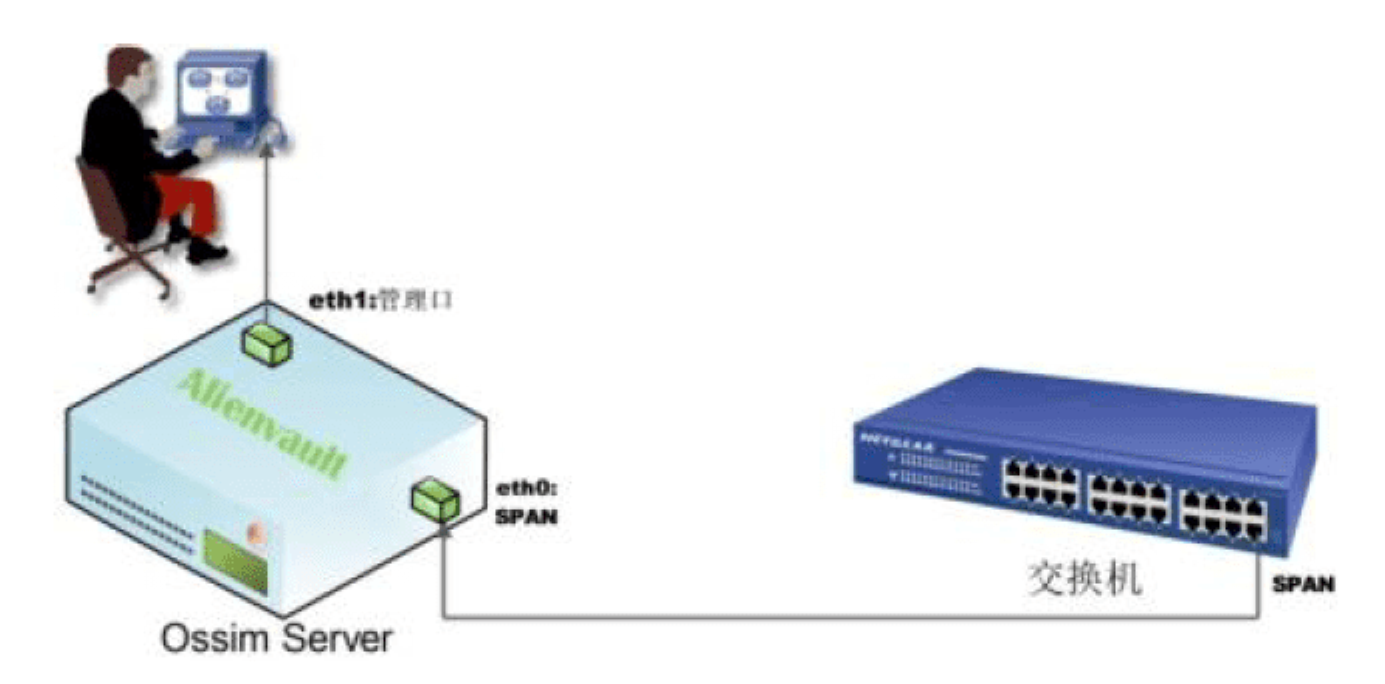


图 2-33　交换机与 OSSIM 连接方式

服务器上采用双网卡，其分工通过 OSSIM 命令行控制台 AlienVault Setup，选择 Configure Sensor→Configure Network Monitoring，选取新添加的网卡并接受应用设置，即可将管理口和监控口网卡区分开。

2.7.3 分布式部署（VPN 连接 ）举例

VPN（虚拟专用网）是指运用加密、认证、访问控制等技术在公用网络上构建的专用逻辑虚拟子网，它使得物理上分布在不同地点的相关用户通过安全的“加密管道”在公用网络中通信。下面看一个 OSSIM 使用 VPN 的实例。首先安装 OSSIM Server 4.1，四个组件均安装，服务器和传感器 IP 规划如下：

- 服务器 IP：92.168.225.20。
- Sensor IP：92.168.225.50。

在服务器器成功启动后，先检查各项服务工作是否正常，然后开始安装传感器，这里我们选用 VPN 方式和服务器连接，在安装时为系统分配一个内网 IP，主要用于和服务器通信。我们输入服务器 IP 地址，这时系统提示“Would you like the system to configure the connection between this host and Alienvault Server（192.168.225.20）using a VPN Network?Network connectivity between the two host will be required to apply this configuration”，如果按往常操作我们不通过 VPN，直接选择 NO 就会继续安装，这里我们选择 Yes 进行 VPN 连接测试，系统提示“Please,enter the root user password of the remote Alienvault Server（192.168.225.20）”。注意：192.168.225.20 这个地址为自己设置。

当 VPN 连接成功以后，系统提示“A VPN Network has been configured between the AlienVault Server and this host. The IP address used by this host within the VPN network is 10.67.68.12,and the ip address of the AlienVault Server is 10.67.68.1”。

1．检查 Server 和 Sensor 之间通信

（1）检查日志。

检查日志命令如下所示。执行结果如图 2-34 所示。

```
sensor:~# tail -f /var/log/ossim/agent.log
```

```
2013-09-17 22:20:31,930 Output [INFO]: event type="detector" date="1379427631" device="192.168.225.2
0" interface="eth0" plugin_id="4003" plugin_sid="16" src_ip="10.67.68.12" dst_ip="192.168.225.20" ds
t_port="22" userdata1="bG9jYWxob3N0" log="U2VwIDE3IDIyOjIwOjMxIGxvY2FsaG9zdCBzc2hkWzE1MTcwXTogQWRkcm
VzcyAxMC42Ny42OC4xMiBtYXBzIHRvIGxvY2FsaG9zdCwgYnV0IHRoaXMgZG9lcyBub3QgbWFwIGJhY2sgdG8gdGhlIGFkZHJlc3
MgLSBQT1NTSUJMRSBCUkVBSy1JTiBBVFRFTVBUISA=" fdate="2013-09-17 14:20:31" tzone="8.0" event_id="1fa411
e3-81b1-000c-299e-51114f7364d0"
2013-09-17 22:20:32,946 Output [INFO]: event type="detector" date="1379427632" device="192.168.225.2
0" interface="eth0" plugin_id="4004" plugin_sid="2" src_ip="10.67.68.12" dst_ip="192.168.225.20" use
rname="cm9vdA==" userdata1="c3NoZFsxNTE3MF0=" userdata2="MA==" userdata3="MA==" userdata4="c3No" log
="U2VwIDE3IDIyOjIwOjMyIGxvY2FsaG9zdCBzc2hkWzE1MTcwXTogcGFtX3VuaXgoc3NoZDphdXRoKTogYXV0aGVudGljYXRpb2
4gZmFpbHVyZTsgbG9nbmFtZT0gdWlkPTAgZXVpZD0wIHR0eT1zc2ggcnVzZXI9IHJob3N0PTEwLjY3LjY4LjEyICB1c2VyPXJvb3
Qg" fdate="2013-09-17 14:20:32" tzone="8.0" event_id="1fa411e3-81b1-000c-299e-5111500e5f94"
```

图 2-34 查看 Agent 日志

（2）插入一个新 Sensor，并进行验证。

在 Configuration→Alienvault Components 选项中插入一个新的 sensor，如图 2-35 所示，IP 地址设置为在 VPN 中指定的 Sensor IP。

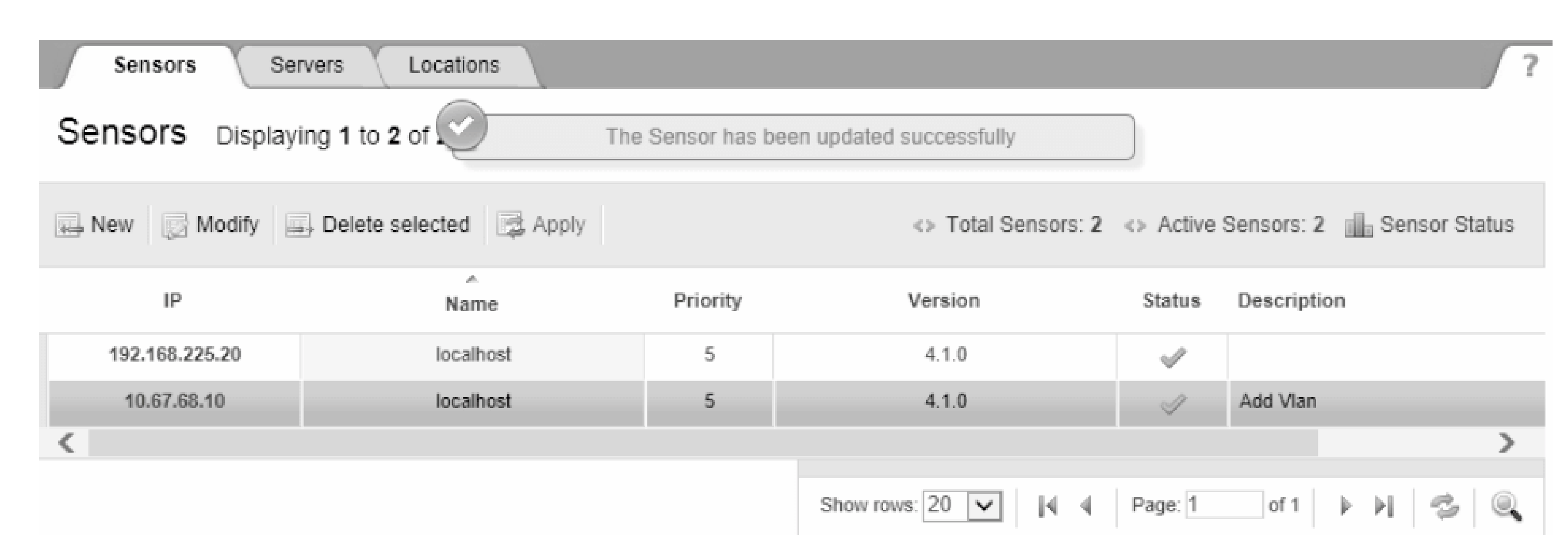


图 2-35　加入新 Sensor

然后，通过 SSH 登录 Sensor 即会产生若干日志。接下来就可以在 Analysis→SIEM 中反映出日志。如图 2-36 所示。

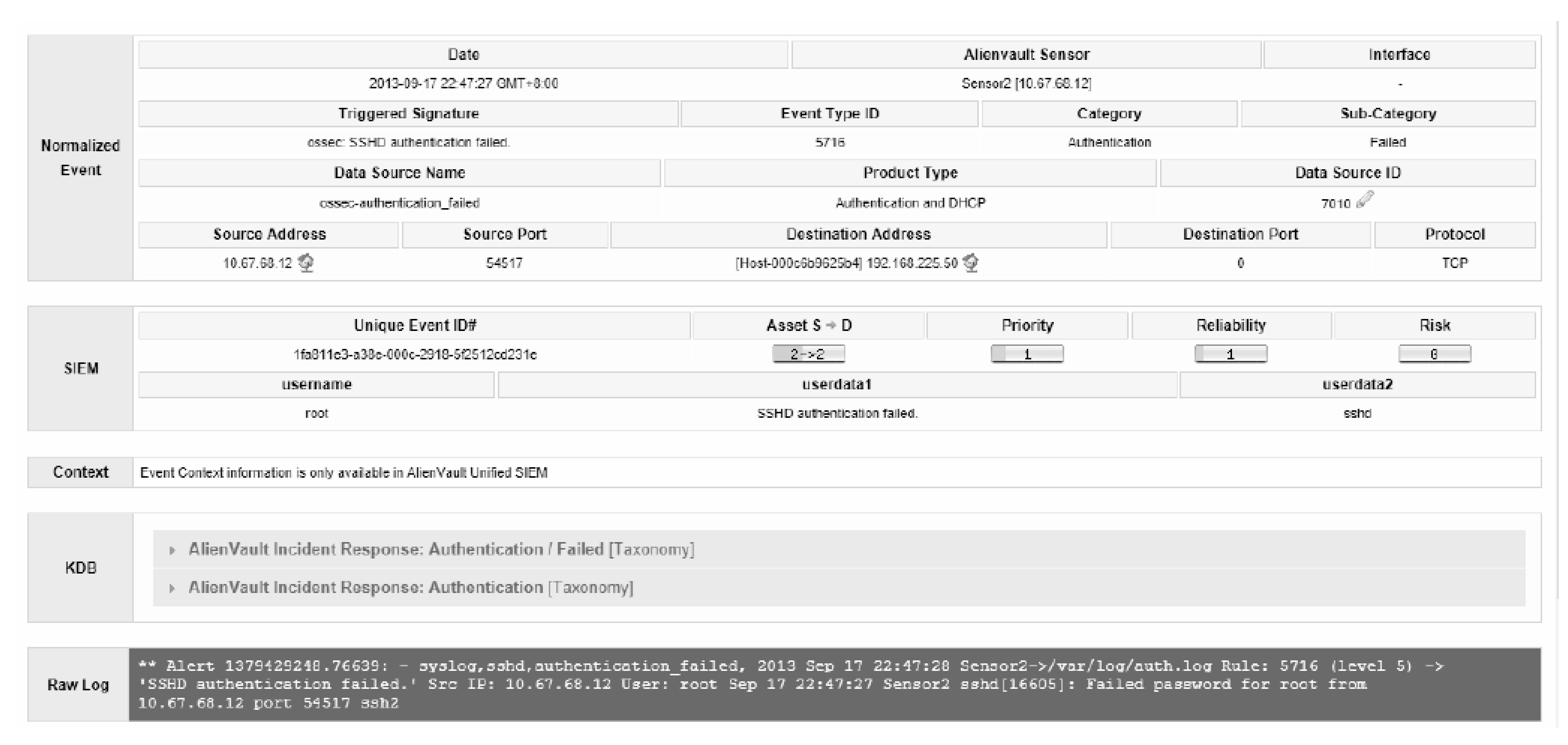


图 2-36　查看日志

在图 2-36 中显示的 SIEM 中，Unique EventID 表示方式为 UUID（有关 UUID 的知识，读者可参考 http://chenguang.blog.51cto.com/350944/1662284），它是数据库中的主健值，能唯一地识别事件。首先确保 Server 端的 OpenVpn 服务工作正常，然后在 Sensor 端，安装时可以选择服务器 IP 和 root 的口令，用来进行 VPN 连接。

2. 需要注意的几个特殊文件

VPN 地址池在 Server 端的 ossim_setup.conf 配置文件 [vpn]中定义，通信端口默认为 33800。在 OSSIM Server 端，/etc/openvpn/AVinfraestructure/keys/目录下有几个证书也需要注意，它们是安装 OpenVPN 过程中生成 Root CA 证书，用于签发 Server 和 Client 证书，其中 ca.crt、ca.key 为根证书文件。

其中 dh1024.pem 为服务器生成 Diffie-Hellman 文件（Diffie-Hellman 是一种确保共享 KEY 安全的方法）。Diffie-Hellman 机制的巧妙在于要求安全通信的双方，可以使用这个方法确定对

称密钥，同样用这个密钥进行加密和解密。

密钥中 alienvcd.csr、alienvcd.crt、alienvcd.key 为 OSSIM 服务器生成的密钥及证书。在服务器端的/etc/openvpn/nodes 目录下存放着节点密钥的压缩包，格式为 IP.tar.gz（例如 192.168.225.50.tar.gz），在 Sensor 端的/etc/openvpn/192.168.225.50/目录下，存放着根证书和“.crt”、“.csr”、“.key”为扩展名的三个证书文件。务必保存好这几个证书，防止根证书文件泄露。注意：这里的 192.168.225.50 为示例 IP 地址。

2.7.4 安装多台 OSSIM（Sensor）

下面实验将带领大家安装分布式 OSSIM 系统。本节实验中 Sensor 配置的内存为 OSSIM Server 的一半，实验拓扑如图 2-37 所示，IP 设置如下：

```
OSSIM ServerIP：192.168.91.130
Sensor1 IP :         192.168.91.131
Sensor2 IP:          192.168.91.135
```

（1）首先安装 OSSIM USM，然后再安装 Sensor。

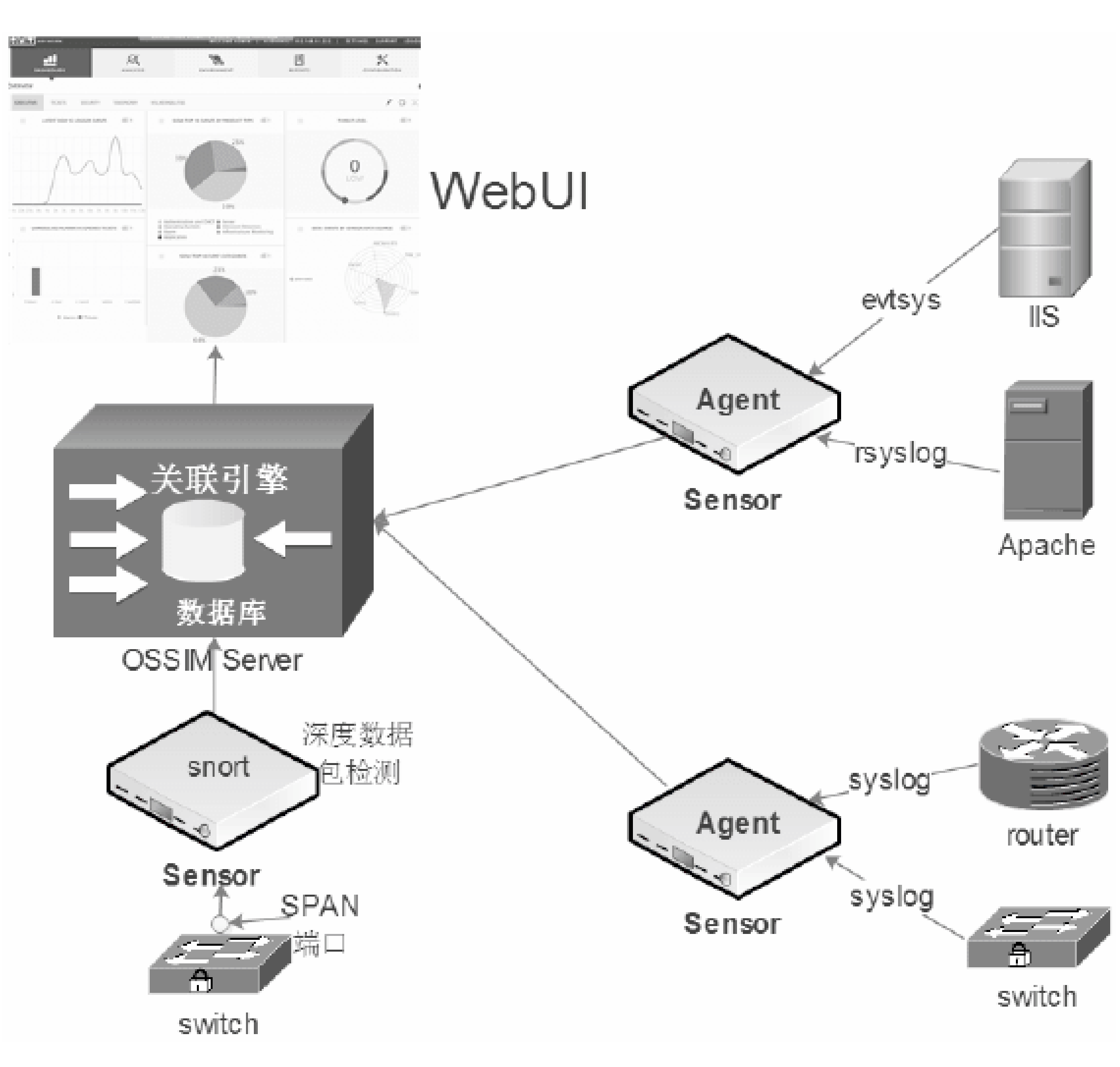


图 2-37 OSSIM 分布式安装

（2）设置 Sensor。

在 OSSIM 4.6 及以上系统添加多个 Sensor 和 OSSIM 4.1 的过程有些不同，我们看看以下步骤：①在 Sensor 的控制台上，设置连接到 Server，具体做法是在 Alienvault Setup 界面下；②选择 Configure Sensor；③选择 Configure Alienvault Server IP；④输入 OSSIM Server 地址；⑤配置 Configure Alienvault Framework IP；⑥输入 OSSIM Server 地址（192.168.91.130）；⑦设置完毕之后返回主菜单；⑧选择 Apply all Changes，使设置生效，如图 2-38 所示。下面需要在 OSSIM Web UI 中继续添加 Sensor 信息到数据库。

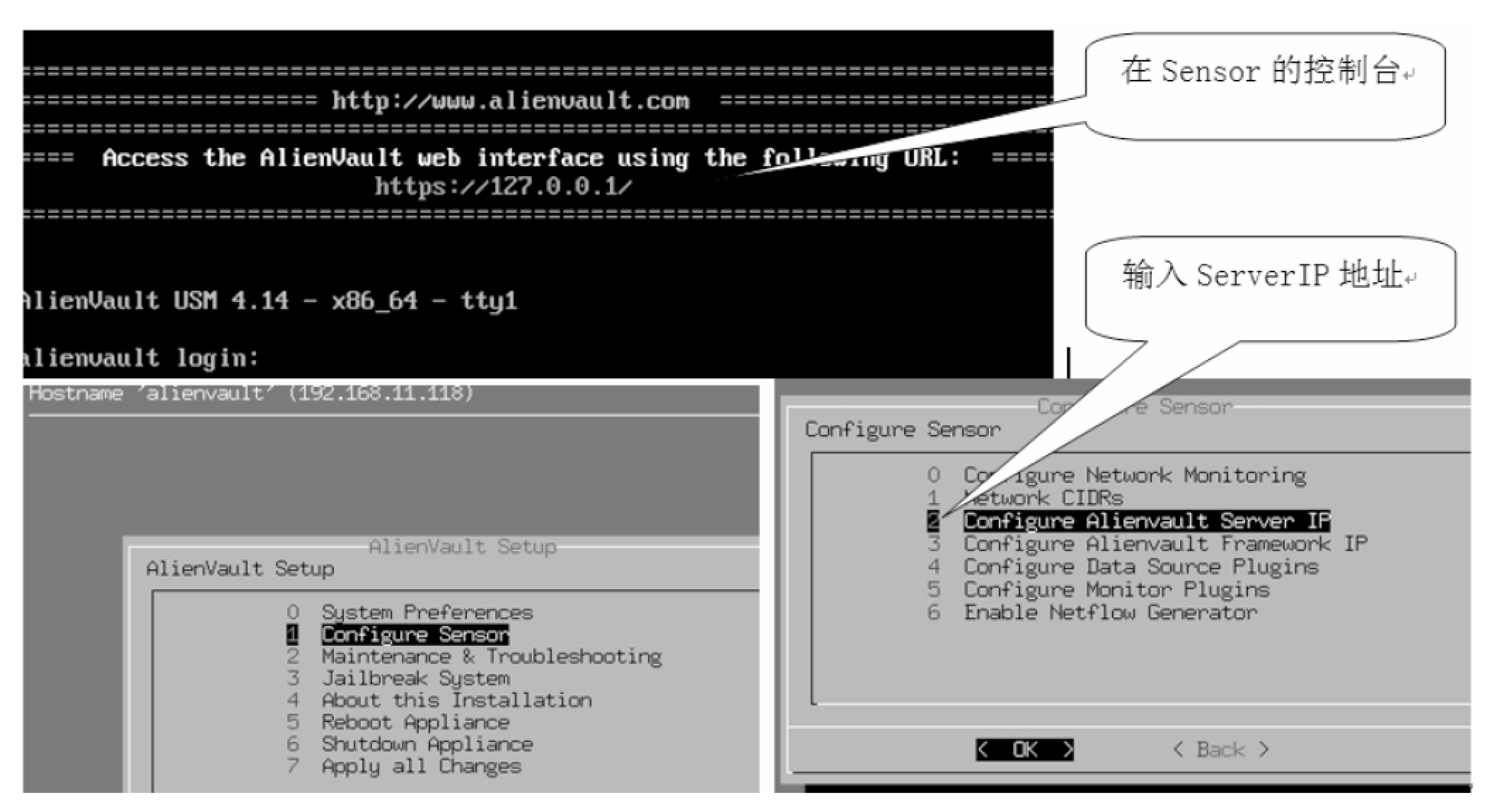


图 2-38　设置 Sensor

（3）在 OSSIM Web UI 上设置连接。

当 Sensor 连接到 Server 并重新启动之后，我们在 Web UI 菜单中依次单击 Configuration→Deployment→Components→Sensors，这时能看到如图 2-39 所示的提示。

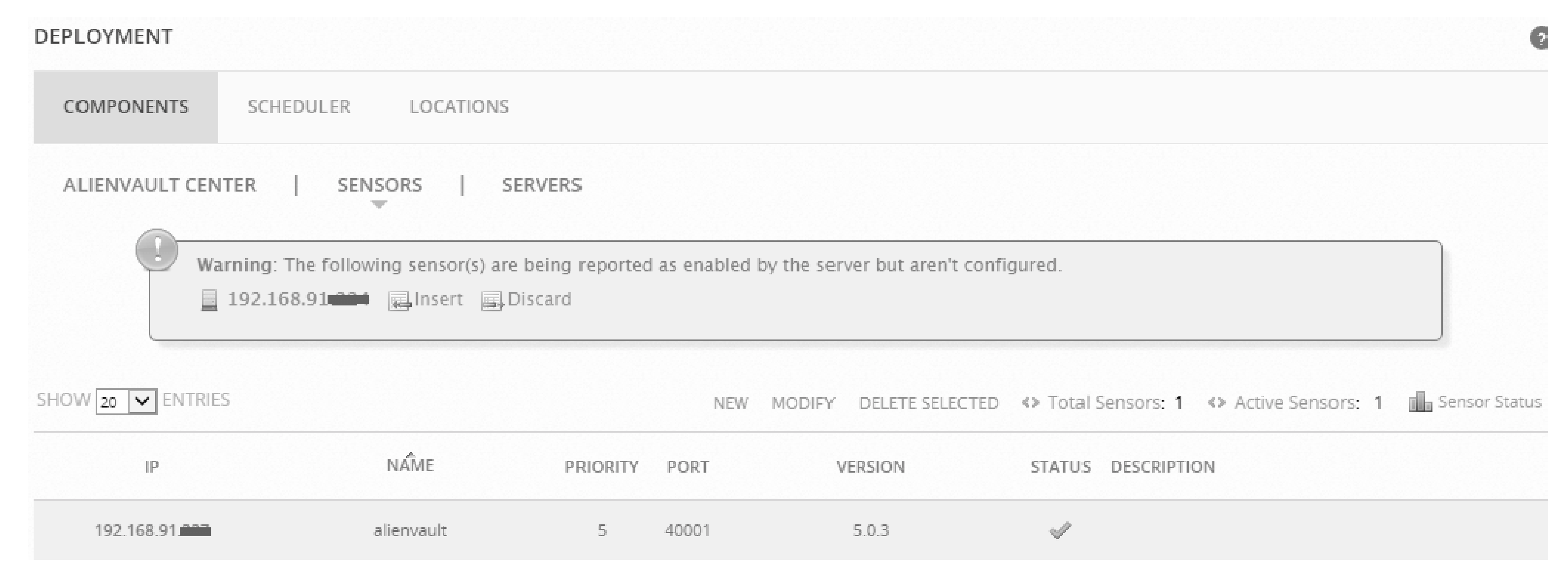


图 2-39　添加 Sensor

这时选择“Insert”按钮，而不能使用选择“New”手工输入 IP 地址的方法。

只有在进行 root 管理员密码验证成功后，才能添加到 OSSIM Server，添加界面如图 2-40 所示。

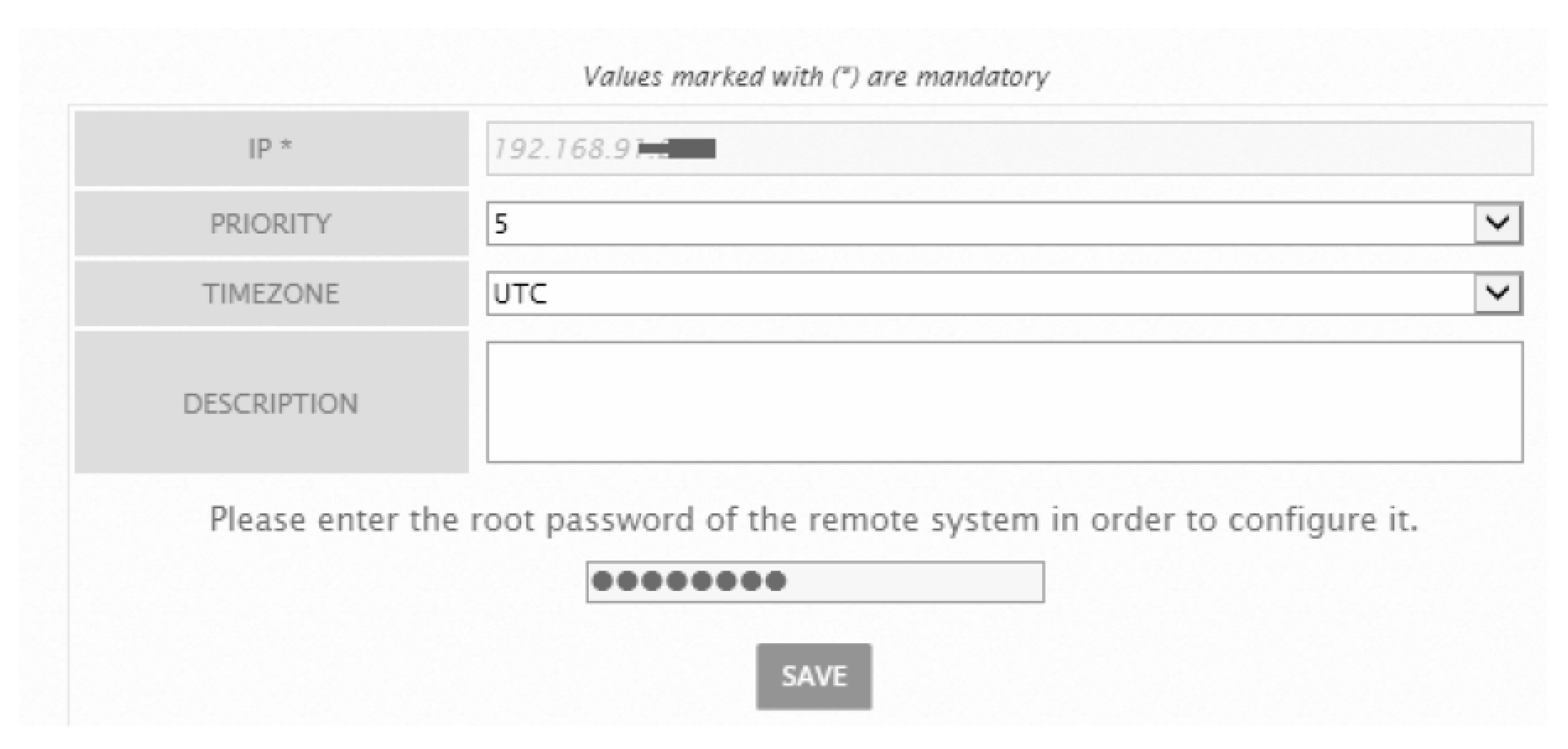


图 2-40　进行管理员验证

经过以上 3 个步骤即可完成 Sensor 与 Server 之间的连接。

（4）为 Sensor 改名。

首次添加的 Sensor 机器名称默认都为 alienvault，这样容易混淆，所以我们必须手动修改各 Sensor 的名称。在 Deployment→Components→Sensors 菜单下，单击新添加的传感器 alienvault，会出现图 2-41 所示画面，这时可在 Name 选项中修改主机名。

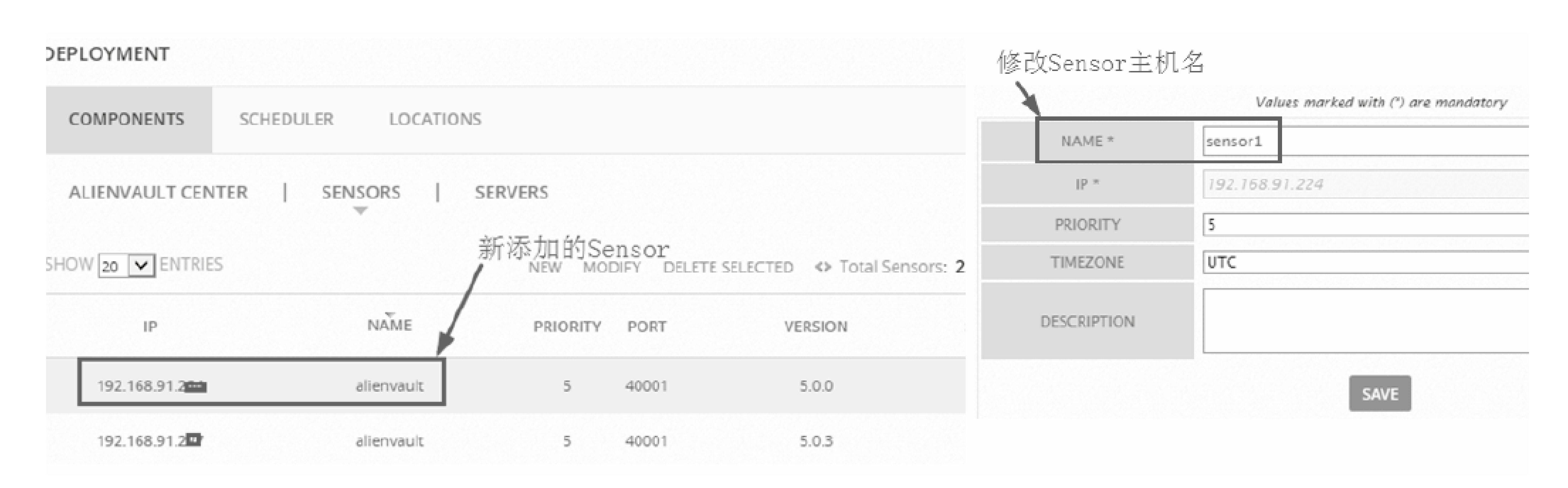


图 2-41　修改 Sensor 名称

Time zone 为 GMT+8（北京时间），整个分布式系统统一为 GMT+8。

（5）连接验证。

在添加 Sensor 成功后，我们可以在多个地方查看传感器：Web UI 首页的仪表盘中查看，如图 2-42 所示。Ntop 服务中查看 Sensor，如图 2-43 所示。

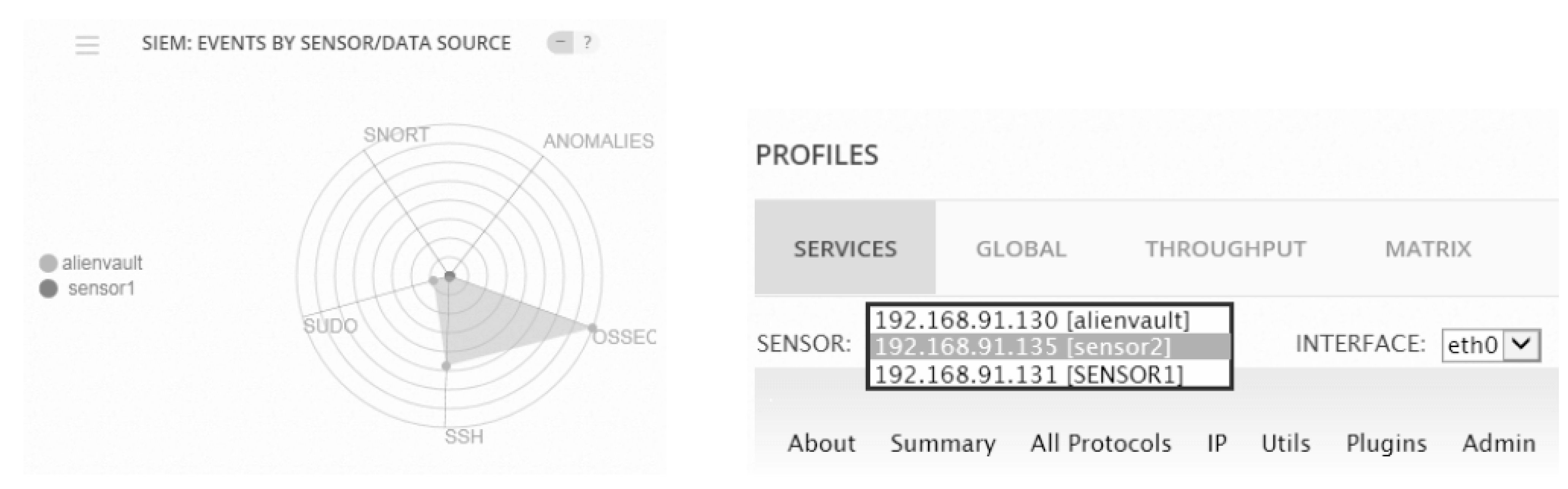


图 2-42　仪表盘查看 Sensor　　　　图 2-43　Ntop 服务中查看 Sensor

在多 Sensor 环境中，需要指定默认 NTOP Sensor，如图 2-44 所示。

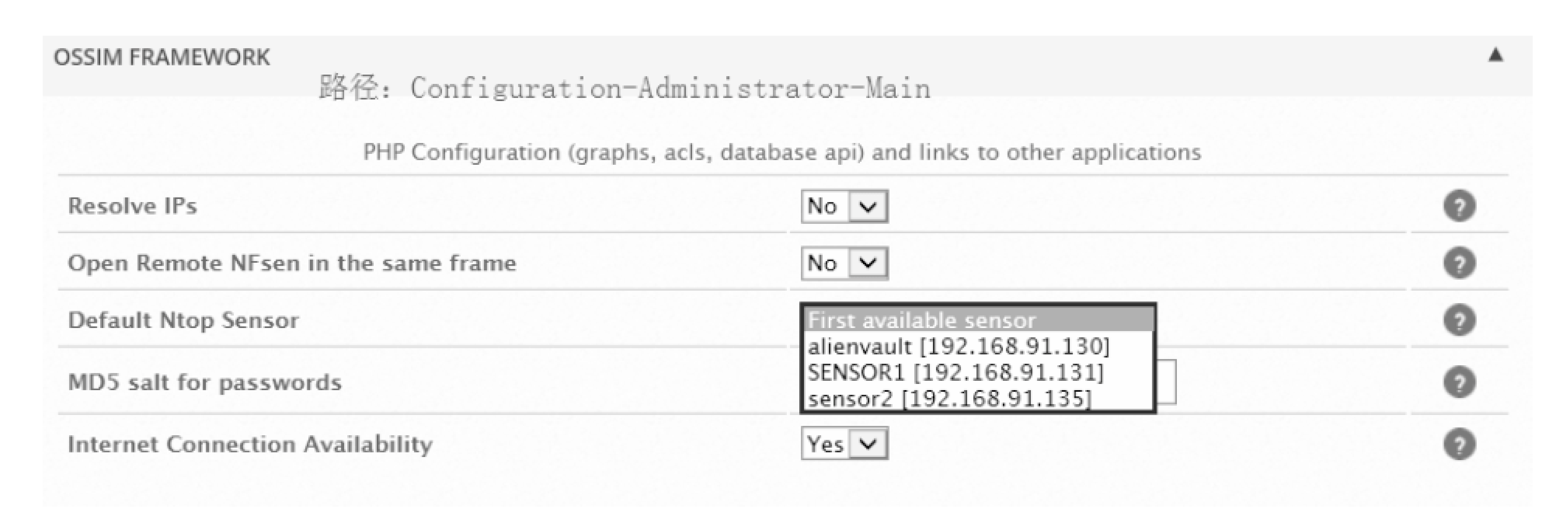


图 2-44　多传感器选择

在多 Sensor 环境中的 SIEM 控制台中，进行高级搜索时，也需要指定对哪一个 Sensor 进行搜索，如图 2-45 所示。

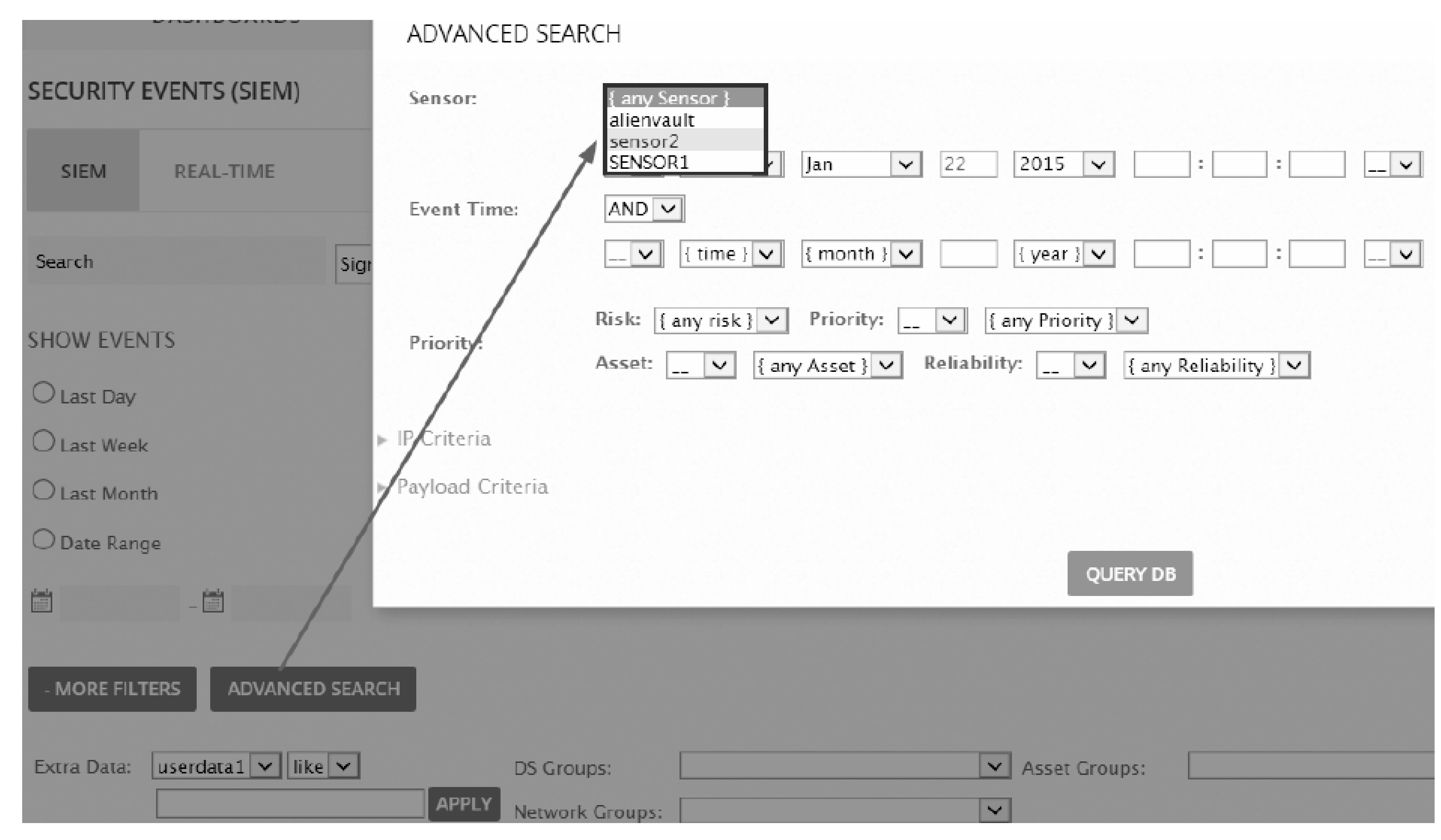


图 2-45　查询不同 Sensor 中的 SIEM 事件信息

（6）中断 Sensor 和 OSSIM Server 的连接。

Sensor 已经成功连接 OSSIM Server A 后，能否再次作为另一台 OSSIM Server 的传感器使用呢？除了新装一台方法之外，我们只需要改变一点配置即可达到目的。首先在 OSSIM Server A 上删除 Server 与 Sensor 的配置连接，然后再到 Sensor 控制台上添加新的 Server 地址和框架地址，接着选择保存并应用系统设置，最终可以到 Web UI 界面中把 sensor 添加进来。

如果直接删除 Sensor 连接文件，系统会提示无法删除，如图 2-46 所示。这时需要删除 Sensor 下的 Asset 和所监控网段的信息，然后 Sensor 连接文件才可顺利删除。

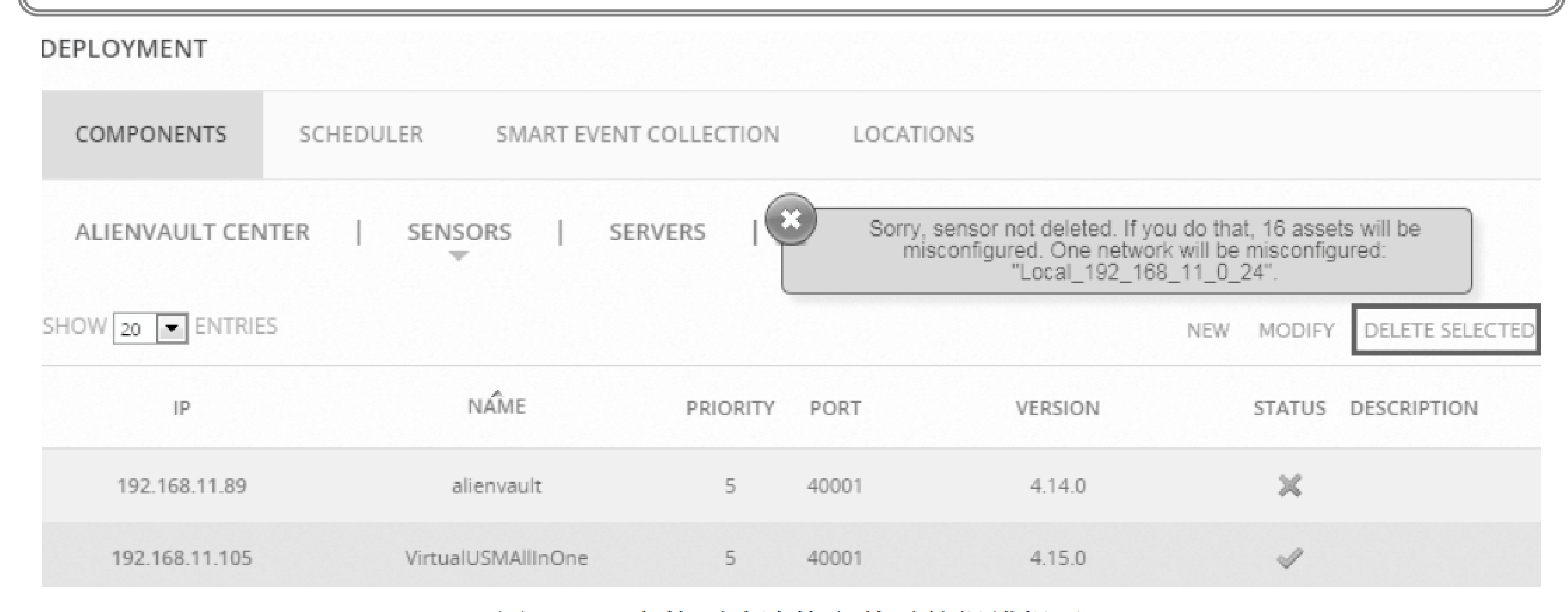


图 2-46　直接删除连接文件时的报错提示

（1）注意 1：如果输入密码不正确，系统提示“Cannot add system with IP 10.32.1X.Y Please verify that the system is reachable and the password is correct.”时可以重新输入正确的密码。如果在图 2-46 中，单击“NEW”按钮来建立一条连接则不会连接成功。

（2）注意 2：当新的 Sensor 添加到 OSSIM Server 之后，便会在/var/alienvault/目录下产生一个以 UUID 命名的目录，这个具体名称就是 Sensor 上执行“alienvault-system-id”的结果。该目录下的 ossec 目录用于存放规则和 ossec agent 可执行文件。

（3）注意 3：当成功添加完成 Sensor 与 Server 之间连接之后，在 Sensor 的/home/avapi/目录下会生成两个特殊目录“.ssh”、“.ansible”，而且在“.ssh”目录下还会生成 authorized_keys 文件，默认权限为 600，sshd 进程会使用该文件中的 key 进行验证。

除了 Web 方式可以直观地显示 Server 和 Sensor 状态以外，还可以通过命令行方式获取信息。

```
#alienvault-api systems
Gathering systems data…
564dc9cd-5bd5-8a3d-d613-3a323a8e4cba - 'trustis-ossim-dmz' - 192.168.101.156 - Reachable
44454c4c-5700-104c-804c-b7c04f543032 - 'trustis-ossim' - 192.168.107.45 - Reachable
```

第二种方法，同样可以获得注册系统的 ID 号：

```
#alienvault-api get_registered_systems
```

第三种方法，直接从数据库中查询：

```
#echo 'select name,inet6_ntoa(admin_ip) from system;' |ossim-db
```

Alienvault Center 和 Sensor 的通信依托 SSH 协议，无须输入 SSH 密码登录，一旦 SSH 证书出现问题，例如我们可以把/home/avapi/.ssh/authorized_keys（此时文件权限为 644）稍微处理一下，那么再次查看 Alienvault Center 状态就会出现“Down”提示，表示双方无法连接。一些初学者好高骛远，最基础的单机部署都没做，直接开始分布式部署，再加之错误的配置方式，容易造成 Server 和 Sensor 之间无法通信，如图 2-47 所示。希望读者从单网卡的混合方式安装，逐渐过渡到多网卡的分布式安装模式。

图 2-47 Sensor 与 Server 中断

如果出现报错，就需要重新添加一次，可在 Sensor 的控制台上使用如下命令：

```
#/etc/init.d/ossim-agent stop
#/etc/init.d/ossim-agent start
```

（4）注意 4：多传感器的选择不适合于 Nagios 服务，在 Environment→Availability→Monitoring 中选择其他 Sensor，则会出现“Unable to connect to Nagios sensor”。在分布式 OSSIM 系统中往往有多个探针，通常我们需要设置首个有效的传感器，具体调整位置在 Configuration→Administration→Main→OSSIM Framework 菜单下面 Default Ntop Sensor 选项中调整。

（5）注意 5：分布式 OSSIM 系统中，Sensor 配置里 OUTPUT 的 IP 地址为 OSSIM Server 地址，在混合式 OSSIM 系统中 Sensor 里 OUTPUT 的 IP 为 127.0.0.1，参考图 1-15 所示。

有些读者或许会思考，到底 OSSIM 支持多少个 Sensor 连接，有限制吗？只要硬件配置足够，对于分布式环境支持的 Sensor 数量没有限制。读者还可以通过在命令行下输入以下命令从数据库来查看添加的 Sensor 主机名和 IP 地址，以及 UUID 等信息。如图 2-48 所示。

```
alienvault:~# echo 'SELECT NAME,inet6_ntoa(admin_ip) FROM system;' |ossim-db
NAME    inet6_ntoa(admin_ip)
alienvault      192.168.91.225
alienvault      192.168.91.224
alienvault:~# _
alienvault:~# alienvault-api get_registered_systems
{"564df90b-6ca3-f88b-7c2b-91ee2d642142": {"admin_ip": "192.168.91.224", "vpn_ip"
: "", "hostname": "alienvault"}, "564d576c-a36a-9f3d-706e-638f61262008": {"admin
_ip": "192.168.91.225", "vpn_ip": "", "hostname": "alienvault"}}
alienvault:~# _
```

图 2-48　查看管理 IP 地址及网络信息

2.7.5　Sensor 重装流程

部署在各 VLan 中的 Sensor 如果发生故障，例如发生宕机事故，将无法收集监控信息，这时需要管理员立即恢复 Sensor，这里的修复就只是重装，因为这种方式速度最快。方法是选择安装菜单中第二项 Sensor，经过 15~20 分钟，Sensor 即可安装完毕。接下来开始配置 Sensor，将它加入 OSSIM Server 中。

首先，升级系统（alienvault-update），确保新添加 Sensor 机器名称和原来相同，否则事件的 Sensor 属性会出现“N/A”，代表 Not Applicable（不适用）。

其次，同步时钟、添加监控网段、添加监控插件。

再次，配置 Sensor→配置 Alienvault Server IP→配置 Alienvault Framwork IP。

最后，所有的配置都和以前故障机 Sensor 一样，配置好后可以到 Configuration→Deployment→Components 的 Sensors 中进行确认。做完这些工作，一台新的 Sensor 又可以继续为 OSSIM Server 收集信息。

2.8　添加 VPN 连接

2.8.1　需求

如果需要监控多个分布在异地的数据中心，则需要在异地的机房内安装 Sensor，最后将 OSSIM Server 和 Sensor 之间通过 VPN 方式连接，以便安全地获取数据。注意以下实验在两台机器中完成。

采用 VPN 方式连接，在 OSSIM 采用 OpenVPN，它基于 OpenSSL 库的应用层 VPN。下面我们以 OSSIM 4.1 为例，假设 Server IP 地址为 10.0.0.30，Sensor 地址为 10.0.0.31，手工配置 VPN 连接方法在下面两个小节中说明。

2.8.2　Server 端配置（10.0.0.30）

（1）编辑文件/etc/ossim/ossim_setup.conf，按以下方式修改变量。

```
server_ip= 10.0.0.30
framework_ip=10.0.0.30
hostname=server
vpn_infraestructure=yes              \\此处默认设置为 no
vpn_net=192.168.1 (为 Sensor 定义 VPN 网络号)
```

（2）为新添加的主机名修改/etc/hosts 和 /etc/hostname。

（3）在修改 ossim_setup.conf 后，必须运行“ossim-reconfig”使得配置生效。

```
server:~# ossim-reconfig -c -v -d
server:~# netstat -nap | grep -i ossim-server
tcp        0      0 0.0.0.0:40001         0.0.0.0:*               LISTEN
16678/ossim-server
```

（4）生成 VPN 配置文件。

首先建立一个 PKI（Public Key Infrastructure，公钥基础设施）。PKI 包括服务端和每个客户端都有一个证书（也称做公钥）和私钥。一个认证机构（CA）的证书和私钥，用来为每一个服务端和客户端颁发证书。

```
server:~# ossim-reconfig --add_vpnnode=10.0.0.31 (sensor IP)
```

此时会在/etc/openvpn/nodes 目录下生成一个 10.0.0.31.tar.gz 的压缩文件，其中保存了 ca.crt、10.0.0.31.key、10.0.0.31.cry 以及 10.0.0.31.conf 配置文件。

接下来，需要将这个压缩包复制到 Sensor（10.0.0.31）主机的/etc/openvpn 目录下。

```
server:~#       scp /etc/openvpn/nodes/10.0.0.31.tar.gz
root@10.0.0.31:/etc/openvpn/
```

2.8.3　配置 Sensor（10.0.0.31）

（1）编辑 /etc/network/interfaces 使用静态 IP ，例如 10.0.0.31。

```
sensor:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 10.0.0.31
netmask 255.255.255.0
gateway 10.0.0.200
sensor:~#/etc/init.d/networking restart
```

（2）在 Sensor 上编辑 /etc/ossim/ossim_setup.conf。

```
admin_ip=10.0.0.31
admin_gateway=10.0.0.200
hostname=sensor
sensor:~# ossim-reconfig - c -v - d
```

（3）添加新主机名到文件/etc/hosts 和/etc/hostname。

（4）解压缩刚才生成的 VPN tar 包，并重启 Openvpn。

```
sensor:~# cd /etc/openvpn
sensor:~# tar -xvzf 10.0.0.31.tar.gz
sensor:~# /etc/init.d/openvpn restart
```

（5）配置 Sensor 上的/etc/ossim/ossim_setup.conf 文件。

```
profile=Sensor
server_ip=192.168.1.1 (vpn server IP)
framework_ip=192.168.1.1 (framework server IP)
detectors=ossec, ssh (enabling ssh and ossec plugins)
sensor:~# ossim-reconfig -c -v -d
```

（6）最后检查 Server 和 Sensor 之间的通信。

检查日志文件：

```
server:~#cat /var/log/syslog | grep openvpn
sensor:~# tail -f /var/log/ossim/agent.log
server:~# tail -f /var/log/ossim/server.log
```

查看 Sensor 显示结果：

```
Configuration -> Alienvault Components and insert sensor with ip 192.168.1.10
(sensor IP in the VPN). After that, try to log into the sensor through SSH to generate
some new events. Then check the results in the GUI under Analysis -> SIEM.
```

Server 和 Sensor 的/etc/ossim/ossim_setup.conf 配置文件中这个配置需要保持一致，即 vpn_infraestructure=yes。

2.9 安装最后阶段

在 OSSIM Server 安装最后阶段会提示“正在运行 cdsetup…”，可能有读者在实验过程中会思考，怎么每次在安装过程最后阶段，即 cdsetup 运行阶段总是很慢呢？是不是在上网更新数据？其实不然，最后结束安装阶段这一过程比较长，主要的工作是创建初始化数据库（alienvault_siem、asec database 及 datewarehouse 等，alienvault-siem 用于记录 Snort 报警事件信息），然后进行各类表创建、插入初始化数据条目，在最后 cdsetup 阶段按 Ctrl+Alt+F4 组合键可以看到：

```
May 11 0103:39 in-target:INSERT ossim-serveres.sql.gz
May 11 0103:39 in-target:INSERT imperva-securesphere.sq..gz
May 11 0103:39 in-target:INSERT ocs-monitor.sql.gz
May 11 0103:39 in-target:INSERT clamav.sql.gz
May 11 0103:39 in-target:INSERT ossim-directive.sql.gz
……
```

以 OSSIM 4.3 版本为例，在 cdsetup 阶段需要创建 13 个数据库，大约 300 多张表和对其进行初始化工作。不过这一步也不是所有安装模式都有，例如前面提到在某个 VLAN 中部署一个传感器 Sensor 就没有必要安装数据库，这时在安装最后就不会创建数据库、表以及写入初始数据。

注意，clamav.sql 和 ossim-directive.sql 等文件内容写到/usr/share/doc/ossim-mysql/contrib/plugins/目录下。我们还需要了解另一个细节，在安装后期将执行/usr/share/alienvault-center/lib/Avconfig_profile_database.pm 脚本，其内容为建立 OSSIM 主要的 13 个数据库，其指令是通过 zcat 直接读取*.sql.gz 压缩包的内容，然后导入到 OSSIM 数据库中。例如：

```
    #zcat /usr/share/doc/ossim-mysql/contrib/OSVDB-tabales.sql.gz | ossim-db
osvdb
```

另一条 OSSIM 中常用到命令如下：

```
    #gunzip <backupfile.sql.gz|mysql  -u 用户 -p 密码 database 数据库名称
```

2.10　OSSIM 安装后续工作

2.10.1　时间同步问题

使用 OSSIM 的过程中，需要用到各种数据分析软件，这些软件需要采集网络上很多的事件信息。“时间戳”对于这些事件及后续的数据分析有很重要的作用。在安装 OSSIM 时，如果用户选错了时区的配置，导致系统时间和当前时间不符，将使取证日志发生偏差。下面介绍如何修改时间。其中 date 命令设置系统时间，而不是硬件时钟。尤其是大家做实验的时候使用虚拟机安装 OSSIM，对于这两个时钟需一致。

我们先要确保时区正确，调整时区命令为：

```
    #dpkg-reconfigure tzdata
```

在弹出对话框中选择“Asia”→“Shanghai”。

当然，也可以通过 tzselect 命令来更改时区，相应的时区文件在/usr/share/zoneinfo/Asia/Shanghai。时区调整完毕之后，继续调整精确的系统时间和硬件时间。

格林尼治时间服务器同步：

```
    # ntpdate time-a.nist.gov
    6 Feb 22:26:54 ntpdate[12159]: adjust  time server 129.6.15.28 offset -0.045589
sec
```

查看时间：

```
    # date
    2014年 12月 22日星期二 12:51:28 CST
```

输入“date -R”可以查看当前时区。

配置硬件时钟：

```
#hwclock --set  --date= "12/13/12 08:30:30"
```

如果系统时钟先前已经设置好，可以根据系统时钟设置硬件时钟：

```
#hwclock  --systohc
```

如果 OSSIM Server 不能接入互联网，使用“data -s”命令手动设置时间和日期。

常用 NTP 服务器如下：

- 上海交大 NTP 服务器：202.120.2.101。
- NTP 上海：ntp.api.bz。

```
#ntpdate  -u 目标 NTP 服务器 IP 地址
```

再加上这两个服务器任何一个即可。

时间同步要求：由于被采集系统的时间不尽相同，无法有效支持审计系统的关联分析、进行事件追查。因此企业网内最好能提供时间同步或者支持采用其他时间同步系统的能力。当具备统一的时间同步服务器时，建议将 OSSIM 系统和被管理系统根据同一时间源信号对时间进行同步。如果没有统一的时间源则通过互联网手动同步。

时间同步操作，默认在 OSSIM 系统安装时就有，但如果系统没有与外网链接，这步骤令后需要管理员手动完成。

2.10.2　系统升级

OSSIM 4.1 可以直接升级到 OSSIM 4.4，但无法直接升级到 4.8 以上系统，需要用新版本 ISO 重新安装。安装完系统后，首要工作是升级软件及补丁，其方法有命令行方式和 Web 界面方式两种。下面对升级的细节做一下说明。

更新源地址：

```
#vi  /etc/apt/sources.list
```

在 sources.list 中优先使用 alienvault 的，我们将 Debian 的源添加在后面：

```
deb http://ftp.debian.org/debian/ squeeze-updates main contrib
deb-src http://ftp.debian.org/debian/ squeeze-updates main contrib
deb http://ftp2.de.debian.org/debian squeeze main non-free
```

此处“non-free”表示安装的软件包含不符合 DFSG（Debian 自由软件指引，Debian Free Software Guidelines）要求。

用命令行方式升级输入以下命令：

```
#service apache2 stop  \\停止 Web 服务
#alienvault-update     \\更新系统，过程较长，切勿强行中断升级
#apt-get upgrade        \\更新已安装的包
```

当执行 alienvault-update 脚本时，首先会在/usr/share/ossim-installer/目录下新建目录 temp 之后，下载 http://data.alienvault.com/RELEASES/alienvault4_update-script、alienvault4_preseed.conf、alienvault4_update-script.sig 三个文件，都可以用文本编辑器查看内容。如果读者想了解升级过程到底修改了那些内容，只要你具备 SHELL 基础即可查看 alienvault4_update-script 脚本。系统会将升级日志存储在/var/log/alienvault/update/目录下便于今后查看。

当升级完成后，再启动 apache2。如果是 OSSIM 4.3 及以上系统，建议使用“alienvault-setup”方式进行升级，在控制台中首先选择“0 System Preferences”，然后选择“5 Update AlienVault System”，如图 2-49 所示。

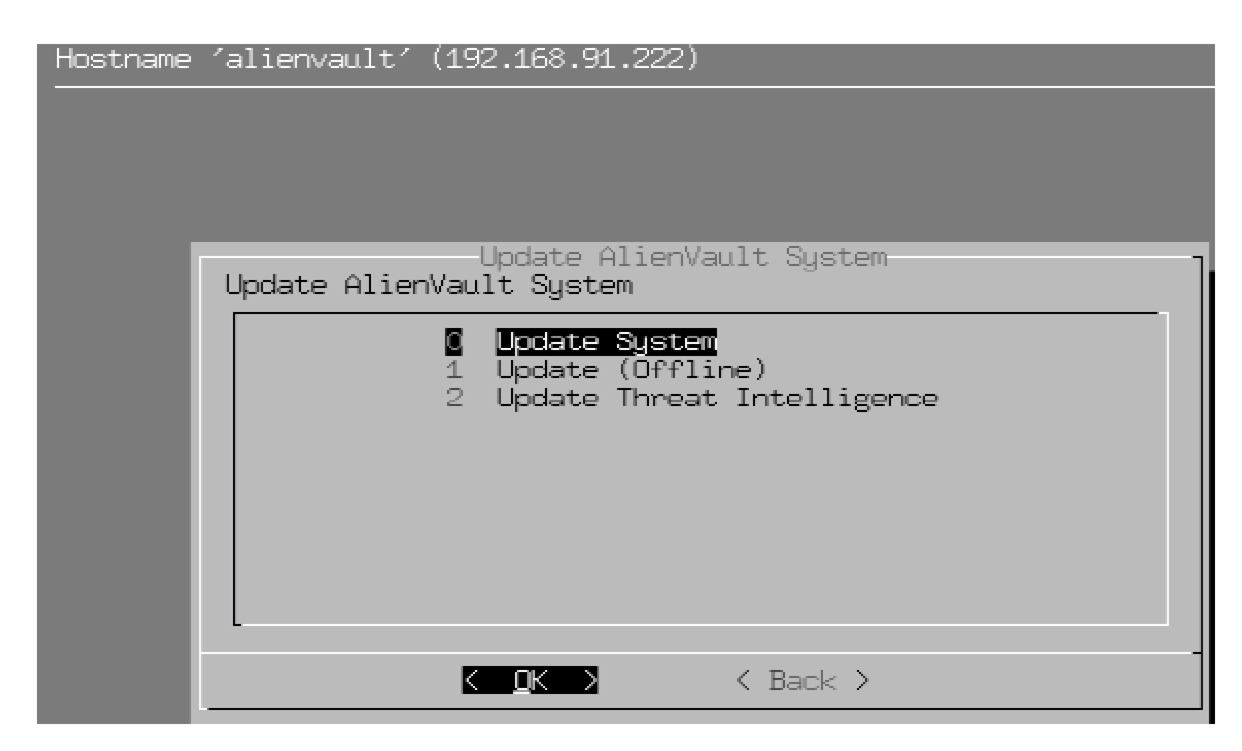


图 2-49　升级 OSSIM

当你再次使用 alienvault-update 更新系统时，很多服务的 conf 配置文件会被覆盖，所以在此操作前一定要备份系统配置文件。

首先更新程序会到下列地址下载数据包。

- http://data.alienvault.com/alienvault4/
- http://security.debian.org
- http://ftp.us.debian.org

以上 3 个站点下载更新包（格式为*.deb），这个更新脚本会自动解决包冲突和软件包依赖问题，它使用 source.list 文件进行软件包管理，在 OSSIM 控制台上，大家可以参考/etc/apt/source.list.d/alienvault4.list 文件。

（1）注意 1：在升级过程中不要强行中断，否则只能通过“dpkg --configure -a”命令修复。手工更新 software_cpe 表。

升级过程比较长，尤其是在“update_percona”、“upgrade_dist_download_onyly”这两个阶段等待的时间更长，此时一定要耐心待系统全部完成后，再继续操作。

如果无法等待系统配置完成就强行终止升级，在以后的包管理中将会发生各种意想不到的

问题，那么只能通过上面介绍的命令尝试修复。

（2）注意 2：OSSIM 3 和 OSSIM 4 的版本路径稍有不同：

OSSIM 3 主要更新包地址为 http://data.alienvault.com/alienvault3/binary/Packages。

OSSIM 4 主要更新包地址为 http://data.alienvault.com/alienvault4/binary/Packages。

系统更新脚本会保存在/usr/share/ossim-install/temp/alienvault4_update_script 文件中。

OSSIM 4.3 以后的版本升级 Server IP 为 70.38.37.7，它对应的域名为 data.alienvault.com。

（3）注意 3：由于 alienvault-update 首次升级系统时间比较长，如果没有耐心等待，强行终止升级，后续安装软件就会遇到问题。有很多 OSSIM 服务和配置脚本会在“ali envault-update”命令执行之后被覆盖，有可能造成以前正常工作的服务在升级后失效，所以建议大家在升级前一定做好配置文件备份及测试环节。

实际上 dpkg 的操作被中断了，必须手动执行“dpkg --configure –a”以修复这个问题，解决方法是输入以下命令：

```
#alienvault_dpkg  --configure -a
```

该过程同样较长，必须耐心等待整个过程执行完成。Web 方式则会在 GUI 界面自动提示待单击确定按钮后会自动进行。更新完毕后，可以通过命令行查看到底有哪些 OSSIM 数据包，使用的命令如下：

```
#dpkg -l |grep ossim
#dpkg -l |grep alienvault
```

当下载完更新包以后，系统升级脚本会调用 dpkg 解压并安装这些包，下面最重要的环节就是 OSSIM Reconfig，它的升级顺序如下：

- Getting IP
- Configuring Server profile
- Configuring Framework Profile
- Set nessus password
- Configuring alienvault-crosscorrelation-free
- Restarting OSSIM-server
- Restarting apache
- Restarting openvas-manager
- Restarting nagios3
- Restarting API service
- Restarting nfsen
- Restarting ossec
- Restarting Asec service
- Restarting ossim-server

- Restarting ossim Framework
- Restarting ossim-agent
- Restarting memcache
- Restarting firewall
- Restarting ntop
- Restarting rsyslog
- Restarting monit
- Restarting rsync
- Restarting fprobe

先是下载然后解包安装，最后重启各项服务结束整个配置过程，回到控制符。在升级过程中不能中断，而且要确保在整个升级过程中互联网出口的顺畅。如果初次安装的是 OSSIM 4.1 版，那么在升级之后会自动升级成最新版本。

如果读者在升级过程中强行退出，将有可能造成系统底层的 dpkg 包管理工具发生错误，万一出现这种情况，大家也别着急，还可以尝试用下列命令修复：

```
#dpkg --configure -a
```

（4）注意 4：因为 OSSIM 的所有升级软件及插件都需要在国外网站同步更新，为了下载保证顺畅，建议升级工作在凌晨进行。在升级过程中出现 Igo、Hit 和 Get 分别代表：

- Igo=Ignored：代表检查被忽略；
- Hit：表示没检查新版本，这意味着目前是最新的包；
- Get: 代表找到了比现在更新的软件版本，需要下载升级，如果上面一行出现 Get XXX，紧接着就会开始下载。

（5）注意 5：一旦 OSSIM 系统最终调试完毕，不建议频繁使用“alienvault-update”命令升级系统，跨版本升级只能重装系统，例如从 3.x 升到 4.x 或 4.x 升到 5.x，这种情况下不建议以 update 方式升级。而且建议先在测试机上完成测试，确定无误后再正式升级。正确的升级方法是在控制台下，通过 Alienvault Setup 界面中的 System Updates 来升级，这样可以避免误操作。

（6）注意 6：OSSIM 系统使用 update 升级后的 deb 所处的位置也需要大家了解，当使用 alienvault-update 命令升级时，首先下载软件包，然后解压并安装，当升级完成后剩下的 deb 文件不会自动消失。如果磁盘空间紧张时，我们可以把这些文件删除，这些文件是 /var/cache/apt/archives/目录下*.deb 文件。

2.10.3　apt-get 常见操作

本书涉及的软件安装都会与 apt-cache 和 apt-get 命令打交道，这两个命令是每位 OSSIM 维护人员必须掌握的操作命令，详细说明如表 2-6 所示。

表 2-6　apt-cache 和 apt-get 命令

操作系统	说明
apt-cache search package	搜索包
apt-cache search ossim	在软件包列表中搜索含有 ossim 字符串的包
apt-cache show package	获取包的相关信息，如说明、大小、版本等
apt-get install package	安装包
apt-get install package - - reinstall	重新安装包
apt-get -f install	修复安装
apt-get remove package	删除包
apt-get remove package - - purge	删除包，包括删除配置文件等
apt-get update	更新源，注意如果在修改了 /etc/apt/sources.list 或 /etc/apt/preferences 之后需要运行该命令
apt-get upgrade	更新已安装的包（在新装的 OSSIM 中常用）
apt-get dist-upgrade	升级系统
apt-get dselect-upgrade	使用 dselect 升级
apt-cache depends package	了解使用依赖
apt-cache rdepends package	查看该包被哪些包依赖
apt-get build-dep package	安装相关的编译环境
apt-get source package	下载该包的源代码
apt-get clean && sudo apt-get autoclean	清理无用的包，清空 apt 的缓存空间
apt-get check	检查是否有损坏的依赖

2.10.4　扫描资产

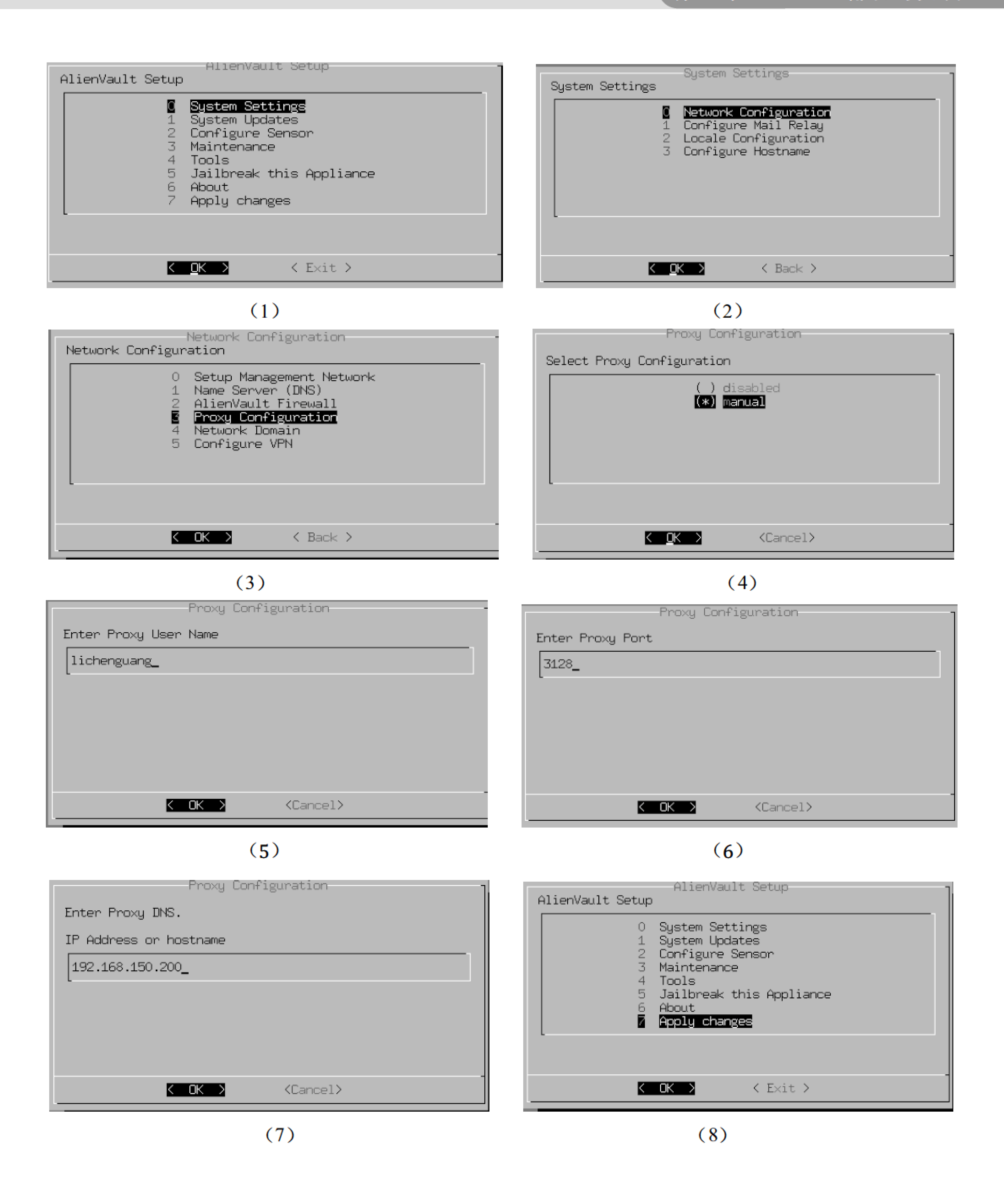
OSSIM 围绕资产进行管理和监控，资产管理在所有服务管理中处于核心地位。作为管理者必须知道你有多少资产，它们在哪里，配置如何，所以在 OSSIM 系统装完后首先开始对监控网段内资产进行扫描，建立资产列表。扫描 Sensor 所监控的网段，建立资产列表可在 Web UI 的 Environment→Assets→Assets Discovery 中输入网段 CIDR，选择 Sensor，然后开始执行，按照默认选项扫描，这时在后台相当于操作远程 Sensor 机器执行了以下命令：

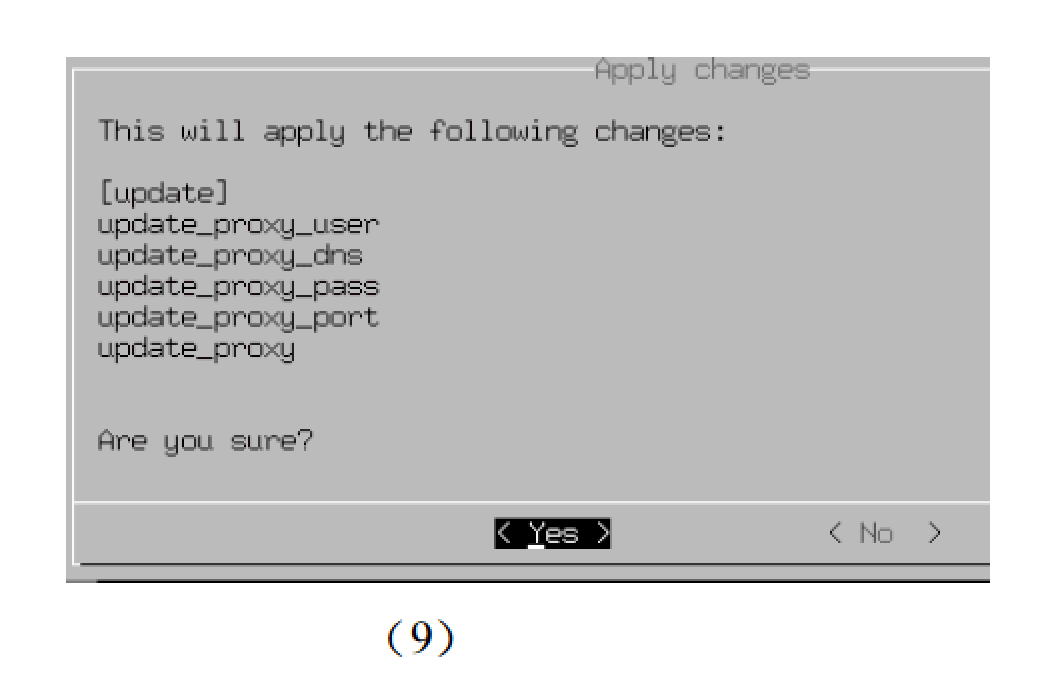
```
#nmap -A -T3 -sS -F x.x.x.x/24  -oX -no-stylesheet
```

不建议用户选择完整扫描，那样将浪费大量时间。获取到扫描机器列表后，继续单击“Update Database Values”按钮，保存到数据库中。

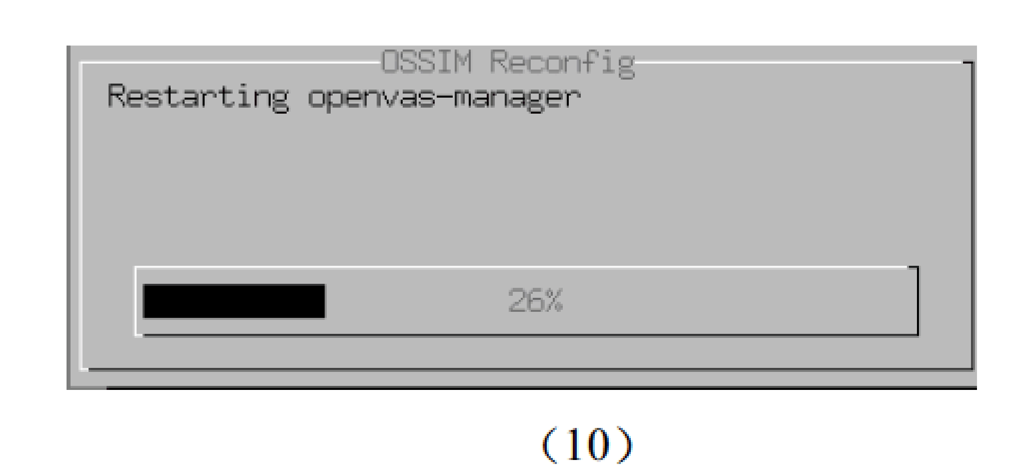
2.10.5　通过代理升级系统

使用代理服务器升级 OSSIM 可以将 OSSIM 系统本身与互联网隔离，以加强系统安全，但系统升级又必须联网进行，NAT 的方式并不安全，所以下面我们通过代理来设置，步骤如图 2-50 所示。


AlienVault Setup
AlienVault Setup
0 System Settings
1 System Updates
2 Configure Sensor
3 Maintenance
4 Tools
5 Jailbreak this Appliance
6 About
7 Apply changes
< OK >
< Exit >
(1)
System Settings
System Settings
0 Network Configuration
1 Configure Mail Relay
2 Locale Configuration
3 Configure Hostname
< OK >
< Back >
(2)
Network Configuration
Network Configuration
0 Setup Management Network
1 Name Server (DNS)
2 AlienVault Firewall
3 Proxy Configuration
4 Network Domain
5 Configure VPN
< OK >
< Back >
(3)
Proxy Configuration
Select Proxy Configuration
( ) disabled
(*) manual
< OK >
<Cancel>
(4)
Proxy Configuration
Enter Proxy User Name
lichenguang_
< OK >
<Cancel>
(5)
Proxy Configuration
Enter Proxy Port
3128_
< OK >
<Cancel>
(6)
Proxy Configuration
Enter Proxy DNS.
IP Address or hostname
192.168.150.200_
< OK >
<Cancel>
(7)
AlienVault Setup
AlienVault Setup
0 System Settings
1 System Updates
2 Configure Sensor
3 Maintenance
4 Tools
5 Jailbreak this Appliance
6 About
7 Apply changes
< OK >
< Exit >
(8)



（9）



（10）

图 2-50　代理设置组图

我们打开/etc/ossim/ossim_setup.conf，观察配置文件变化。当然，我们也可以直接修改配置文件，然后输入“ossim-reconfig”。接着就可以输入“aplienvault_apt-get update”开始通过代理升级系统。

2.10.6　防火墙设置

OSSIM 默认 Iptables 防火墙是打开状态，可以通过以下命令，查看防火墙规则。

```
#iptables -L |more
```

这些规则都记录在/etc/ossim_firewall 这个配置文件里，熟悉 Iptables 的话，可以在这个文件中直接修改。如果在启动系统时不启动 Iptables，可以修改 ossim_setup.conf 配置文件中的[firewall]，将“Active=yes”改成“active=no”。

对于 Iptables 的基础知识，大家可参考我的这篇博文 http://chenguang.blog.51cto.com/350944/1601462。

2.10.7　让控制台支持高分辨率

在高分辨率下，每屏所展现的系统输出信息更多，所以我们希望得到更高的分辨率。默认 Kernel 的 TTY 分辨率有限，解决的方法是给 Kernel 传递 VGA 参数，OSSIM 刚装好时默认分辨率为 800×600 ，我们通过以下命令修改。

```
#vi /boot/grub/grub.cfg
```

将 vga=788 改为“791”。

- vga=788：代表分辨率 VESA framebuffer console @ 800 × 600 × 32k。
- vga=791：代表分辨率 VESA framebuffer console @ 1024 × 768 × 32k。

如果希望还原以前设置，执行如下命令：

```
#update-grub
```

在 OSSIM 4.11 之后的系统，默认为高分辨率控制台。

2.10.8　手动修改服务器 IP 地址

手动修改 IP 地址适合需要修改 OSSIM Server/Sensor 地址的用户，一般用户无须修改。步骤如下:

（1）vi /etc/network/interfaces

改过之后：

```
/etc/init.d/networking restart
```

（2）vi /etc/ossim/ossim_setup.conf

修改第 1 行，admin_ip 字段的 IP。

修改第 40 行，framework_ip 字段的 IP。

改过之后保存退出。

```
#ossim-reconfig
```

（3）修改防火墙规则

```
#vi /etc/ossim_firewall
```

如“-A INPUT -p tcp --dport 3000 -s 192.168.0.10 -j ACCEPT”。

另外，大家还可以在 OSSIM Server 的控制台上，通过输入 ossim-setup 命令启动一个图形化菜单进行修改。

2.10.9　修改系统网关和 DNS 地址

解决方法和上一小节修改服务器 IP 地址类似，修改/etc/ossim/ossim_setup.conf 文件中 admin_gateway、admin_dns 变量的值即可，改过后保存退出，执行以下命令以便设置生效：

```
#ossim-reconfig
```

2.10.10　更改默认网络接口

由于特殊需要，需要将默认 eth0 改成 eth1 接口，下面给出步骤：

（1）添加第二块网卡 eth1。

（2）根据/etc/snort 目录下的 eth0 网卡配置文件 snort.eth0.conf 生成 snort.eth1.conf 做适量调整。

（3）编辑/etc/ossim/ossim_setup.conf，在[sensor]中修改 interface=eth1。

（4）编辑/etc/ossim/agent/plugins/snortunifed_eth0.cfg，并将它的名字改成 snortunifed_eth1.cfg。

（5）运行 ossim-reconfig。

2.10.11 消除登录菜单

在 OSSIM 4.3 版本之后，为了保证安全，加入了登录菜单就是我们看到的 ossim-setup，习惯了命令行操作的朋友感觉这种改变不方便，解决方法是修改/etc/passwd 中 root 所在行中的 shell，具体方法是：

```
#vi /etc/passwd
```

找到 root 所在行，将/usr/bin/llshell 修改为/bin/bash 即可，经过这样修改之后，再使用 WinSCP 就能方便地与 OSSIM 相互传输文件。

注意：如果在 OSSIM 4.8 和 OSSIM 4.9 系统中，修改如下:

```
#vi /root/.bashrc
```

注销如下语句（如图 2-51 所示）：

```
# ~/.bashrc: executed by bash(1) for non-login shells.
#if [ "$jailbreak" != "yes" ];then
#if [[ $- =~ "i" ]];then
#ossim-setup
#exit
#fi
#fi
```

图 2-51 注销.bashrc 配置中的语句

修改后保存退出即可。

2.10.12 进入 OSSIM 单用户模式

默认情况下 OSSIM 无法进入单用户模式，当磁盘故障时则无法修复文件系统。大家可以在刚装完系统时设置单用户登录模式，方法详见 2.14.2 节。

另外在某些情况下 WinScp 无法登录 OSSIM 系统，这时读者可以打开 OSSIM Server 上的/etc/passwd 文件，修改 root 用户的 shell，将原来“/usr/bin/llshell”改为“/bin/sh”即可解决。还有另一种方法，修改.bashrc 脚本，以 root 用户角色修改/root/.bashrc 文件，把以下语句注销即可：

```
#if [ "$jailbreak" != "yes" ];then
#if [[ $- =~ "i" ]];then
#ossim-setup
#exit
#fi
#fi
```

2.10.13 定制系统启动界面

开机引导时动画效果由 Plymouth 来管理，如果你只是要简单地修改一下开机时的背景 Logo，比如关机中出现的 Logo 就在下列地址：

/usr/share/Plymouth/themes/alienvault/alienvault_logo.png，替换即可。

下面通过定制 plymouth 来修改。

（1）安装主题

```
#apt-get install Plymouth-theme*
```

（2）查看当前 plymouth 的主题

```
# plymouth-set-default-theme
```

系统中已经有的主题：

```
# plymouth-set-default-theme -l
Alienvault
Details
Text
```

（3）改变主题

```
# plymouth-set-default-theme -R themename
```

其中选项“-R”要求重新制做 initrd 文件，没有这个选项，主题改变不会生效，重启系统即可以看到效果。

2.11 OSSIM 启动与停止

一般情况下一个 OSSIM Server 服务器端，若干个 Sensor（传感器）组成一个简单的分布式系统。

以 OSSIM 4.3 启动过程为例：

系统启动时，按 F2 键可查看各项服务加载详情。

（1）开机启动

```
GRUB 引导装入内核；
进入 init 2启动级；
启动 monit；（该服务起到监控系统进程，自动重启已经停止运行的程序，和 Watchdog 的功能相似。）
启动 rsyslog；
启动 ACPI 服务；
启动 openvas 扫描器为 openvassd；
启动 openvas 管理器为 openvasmd；
启动 Web Server；
启动 cron；
启动 fprobe；
启动 ipmievd；
启动 SMP IRQ 均衡；
启动 ha_logd；
启动 Memcache；
启动 muni-Node；
启动 Mysql；
启动 nagios；
```

```
    启动 nfsen;
    启动 ntop;
    启动 VPN;
    启动 OSSEC HIDS;
    启动 ossic-maild、ossec-execd、ossec-analysisd、ossec-logcollector、
ossec-remoted、ossec-syscheckd  ossec-monitord;
    启动 ossim agent;
    启动 ossim framework;
    启动 ossim server;
    启动 alienvault idm server;
    启动 postfix;
    启动 squid;
    启动 openbsd secure shell sever;
    启动 the system activity data collector: sadc;
    启动 ipmievd
```

在 OSSIM 4.6 的后续版本中增加了 rabbitmq-server、redis-server、celerd、celerybeat 服务。

（2）系统关闭过程

OSSIM 的关闭过程是，首先关闭 OSSIM Sensor 主机，然后关闭 OSSIM Server。具体方式是在控制台上选择第六项菜单“Shutdown Appliance”，如图 2-52 所示。因为这样操作可以保护数据和文件系统的完整性。

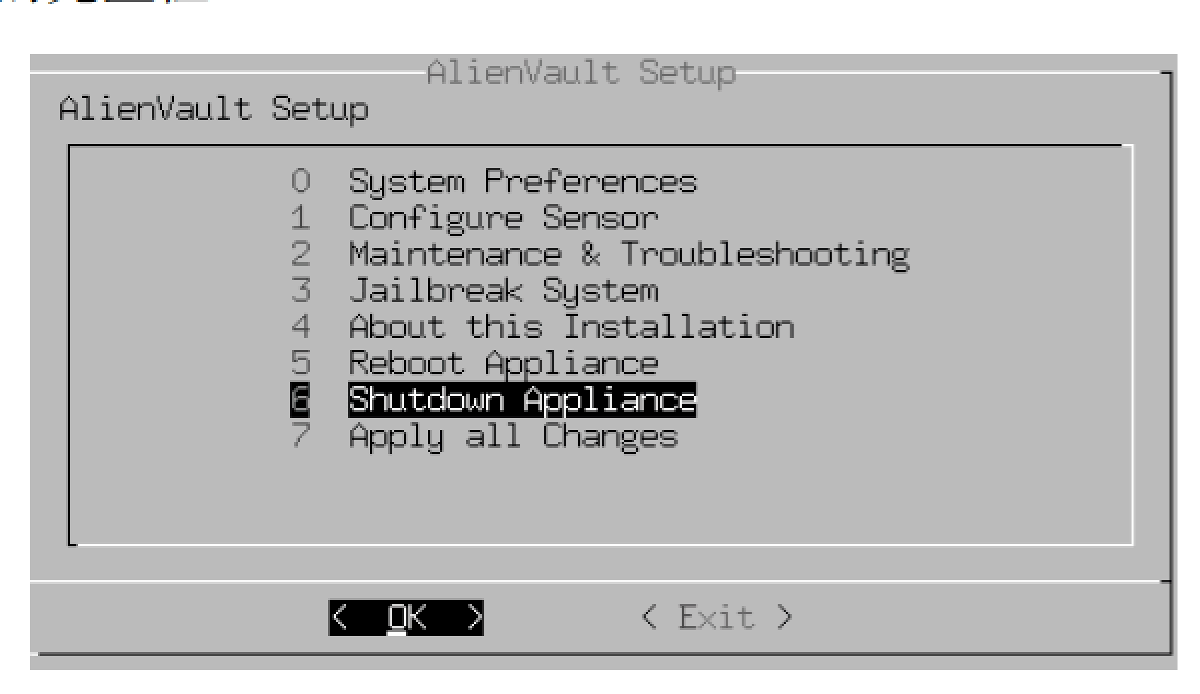


图 2-52　关闭 OSSIM

INIT 进程切换到运行级别 6 发送终止信号以后，系统活动如下：

```
    停止 alienvault  idm server:alienvault-idmstart-stop-daemon;
    停止 fprobe 服务;
    停止 IPMI 事件 daemon     ipmievd;
    停止 ha_logd;
    停止 memcached:memcached;
    停止 web server: apache2;
    停止 daemon 监视程序:  monit;
    停止 Munin-Node;
    停止 nagios3;
    停止 ntop 服务;
    停止 openvas 管理端:  openvasmd;
    停止 openvas 扫描器: openvassd;
    停止 VPN 进程;
    停止 ossec-monitord   ossec-logcollector ossec-syscheckd  ossec-analysisd
```

```
OSSEC HIDS;
    停止 Nfsen  nfcapd;
    停止 OSSIM Agent  ossim-agent;
    停止 OSSIM Server : ossim-server;
    停止 Postfix 邮件服务器;
    停止 resolvconf;
    停止 squid 服务;
    停止 mysql 数据库;
    停止 rsyslog 服务。
```

2.12 安装远程管理工具

默认情况下 OSSIM 系统能够通过 SSH 方式远程管理，下面介绍两款更直观便捷的工具——基于 Web 的管理工具：Webmin 和 phpMyAdmin。

Webmin 是管理员理想的远程管理工具，它通过 HTTPS 协议在保证安全的前提下提供简单而深入的远程管理功能，利用图形化界面引导用户完成配置工作，包括以下主要内容：

- Apache、SSH、Firewall 等各种服务、应用配置及备份任务。
- 系统用户管理，启动项管理，文件系统备份，日志文件轮询配置，PAM 认证管理，计划任务管理，软件包的管理。
- Apache、MySQL、Postfix 等重要服务管理。
- 带宽监控、防火墙配置、TCPWrapper 配置，还包含硬件配置和集群管理。

2.12.1 安装 Webmin 管理工具

OSSIM 支持 Webmin，系统安装此管理工具的目的是方便使用者管理系统。具体安装步骤如下：

（1）在 Webmin 的官网（www.webmin.com）下载 Webmin 安装包（目前最新 1.76），解压并安装（./setup.sh），过程略。

（2）当 Webmin 安装好后，系统将在 10000 端口监听请求，登录系统在浏览器地址栏输入 http://主机名（或 IP）:10000，例如 http://alienvault:10000/。

```
    #netstat -na|grep 10000 \\测试服务是否启动
```

由于 OSSIM 默认没有图形界面，这时会发现本机无法登录，利用基于文本的浏览（lynx）可以测试是否连上系统，在 OSSIM 里安装 X-window 可以连接。建议大家修改 /etc/webmin/miniserv.conf 配置文件，在这个配置文件的最后一行加入“allow=网络号（IP 地址也可以）”，即可实现远程访问。

2.12.2 安装 phpMyAdmin

自从 1998 年 9 月，由 Tobias Ratschiller 发布第一个 phpMyAdmin 工具以来，该项目已成为 MySQL 数据库维护的主要工具之一，它的最大特点就是直观，几乎所有的内容都是通过图形化方式展现，特别是其中分析数据表功能可以帮助我们分析 OSSIM 数据库，由于 OSSIM 系统主要采用了 MySQL 数据库，在对 MySQL 的管控方面比 Webmin 更强大 所以使用 phpMyAdmin 来远程监控和管理数据库比较方便。

例如，当 OSSIM 系统“吃紧”时，它可以帮助我们快速查看服务器运行状况，这样可以迅速地排除故障原因。而且只需单击鼠标就能方便地备份和恢复数据库，不足之处是无法备份数据库中的某几个表，这里告诉大家一个解决办法，我们知道备份出来的都是 SQL 脚本，我们获得备份的文件后打开脚本，找到需要表生成的备份，复制到另一个文本文件并保存为 SQL 扩展名，然后用查询分析器执行即可。

然后执行下面 4 个步骤：

（1）在 http://www.phpmyadmin.net/网站下载 phpMyAdmin 压缩包，实验中下载的压缩包名称为 phpMyAdmin-4.4.14.1-all-languages.zip，代表多国语言版，这也是目前比较新的版本，将它下载到/root 目录，然后解压，进入 phpMyAdmin 目录，将 config.sample.inc.php 修改成 config.inc.php，其他内容不变。

```
#unzip phpMyAdmin-4.4.14.1-all-languages.zip
#cd phpMyAdmin-4.4.14.1-all-languages.zip
#copy config.sample.inc.php config.inc.php
```

（2）将 phpMyAdmin 目录移动到 OSSIM 网站根目录，即/usr/share/ossim/www，再到/etc/ossim/framwork 目录下查看 ossim.conf 配置文件，如图 2-53 所示。

```
#mv /root/ phpMyAdmin-4.4.14.1-all-languages /usr/share/ossim/www
```

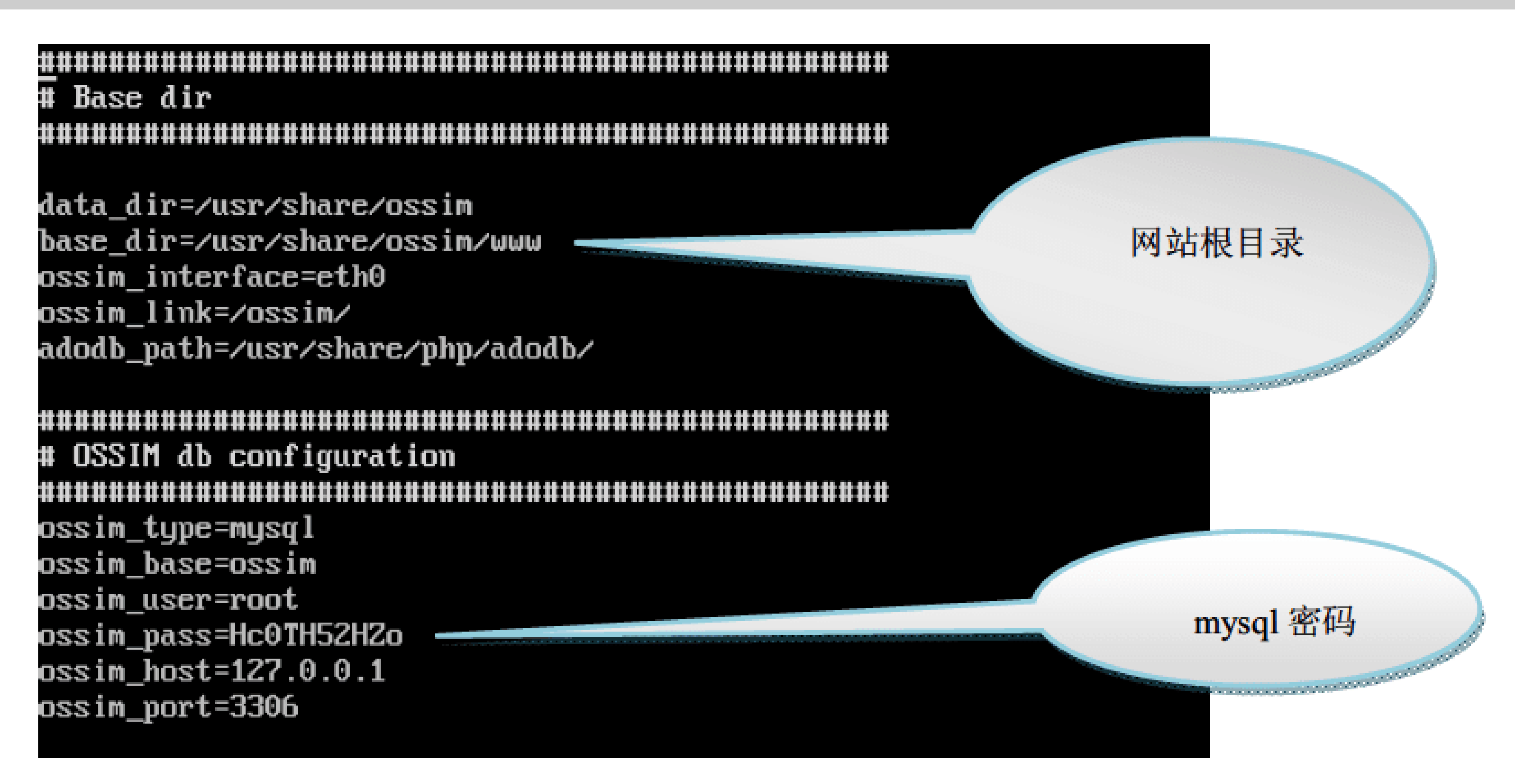


图 2-53 数据库密码

（3）打开浏览器访问 phpMyAdmin，输入网址：https://ip/ossim/ phpMyAdmin-4.4.14.1-all-languages/，提示用户名和密码，有些读者可能不知道如何处理，其实登录用户名为 root，登

录密码为 MySQL 数据库管理员的密码。该密码信息在/etc/ossim/ossim_setup.conf 同样能看到。访问效果如图 2-54 所示。

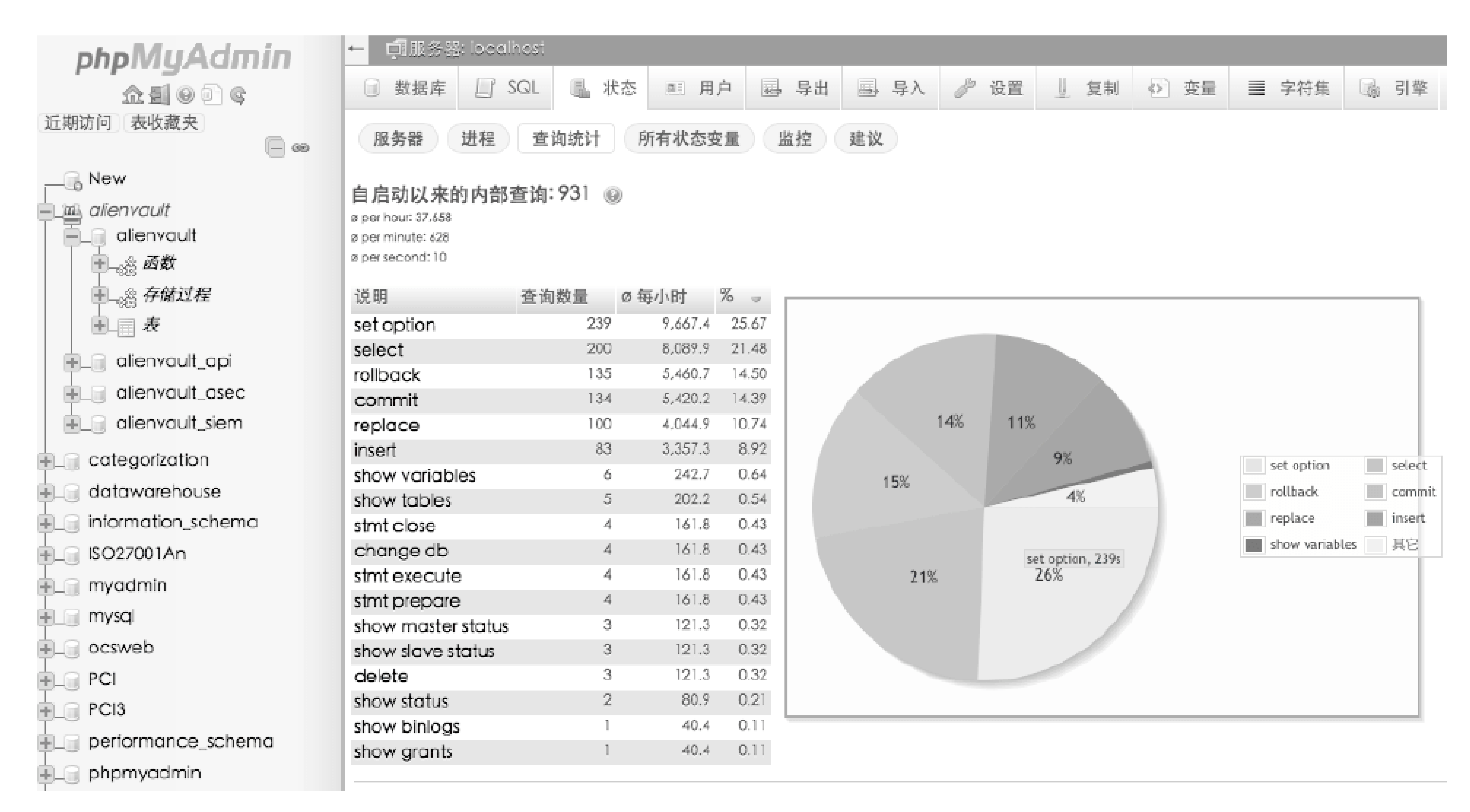


图 2-54 phpMyAdmin 界面

（4）修改登录超时时间：修改 config.default.php 文件中“$cfg['LogincookieValidity']=1440”一行的数值 1440，与此同时还需要确保这个修改的值等于 php.ini 文件中“session.gc_maxlifetime=1440”这行配置中定义的数值。

OSSIM 4.3 系统中默认有 13 个数据库，在左边一栏中共显示了 15 个数据库，后面括号中的数字代表表的数目。

上面讲了源码安装 phpMyAdmin，也可以在 OSSIM 4.x 系统中采用以下命令方式安装 phpMyAdmin：

```
#alienvault-update
#apt-get install phpmyadmin  \\版本为3.3.7，下载包大小约为17MB
```

当见到“Configure database for phpmyadmin with dbconfig-common”提示，选择下一步，此时输入 MySQL 数据库管理员口令，然后设定 phpMyAdmin 口令，接着设置安装在哪一个 Web Server 之下，系统选择 apache2 继续，选择下一步，在这里最容易出现的错误是数据库的权限错误，例如屏幕提示“ERROR 1045（20000）:Access denied for user 'root'@'localhost'（sing password:YES”。在随后出现的对话框中选择“ignore”

系统提示输入数据库和管理员密码，登录后，经过配置就可以使用，这里和源码包使用不同的是，输入地址为 Http://IP/phpmyadmin/。

在使用 phpMyAdmin 时需要注意两点：①安装最新的正式发布版本，这样可以使用 MySQL5.6 的新特性。②安装到服务器后，目录不能保留默认的名称，建议将目录改名。

2.12.3 用 phpMyAdmin 同步功能迁移数据库

在 phpMyAdmin 中使用同步功能迁移数据库，如图 2-55 所示。同步页面会显示源数据库和目标数据库，需要设置源数据库 IP、用户、密码，然后选择本地数据库名称。

源数据库指的是用户需要导出的数据库，目标数据库指的是要导入的数据库，所以，源数据库要选择“远程服务器”，目标数据库就要选择当前服务区器。

图 2-55 迁移数据库

2.13 分布式系统查看传感器状态

在 OSSIM 系统分布式部署中，我们通常需要快速预览多个传感器的各项状态，例如 IDS、漏洞扫描、Netflow 等子系统的工作状态。完成下面实验之前，请确保浏览器能够正常连接谷歌地图，设置方法如下。

2.13.1 设置指示器

首先在 Dashboards→Risk Maps 中定义传感器，首次进入时单击“Set indicators”按钮，为新指示器输入名称之后，在 Assets 资源池中选择一个资源，并在其中选择新建一个 Indicator（指示器），同时在右侧地图中会出现一个图标，如图 2-56 所示，设置新指示器的名称为 test，设置完成后，单击保存按钮。

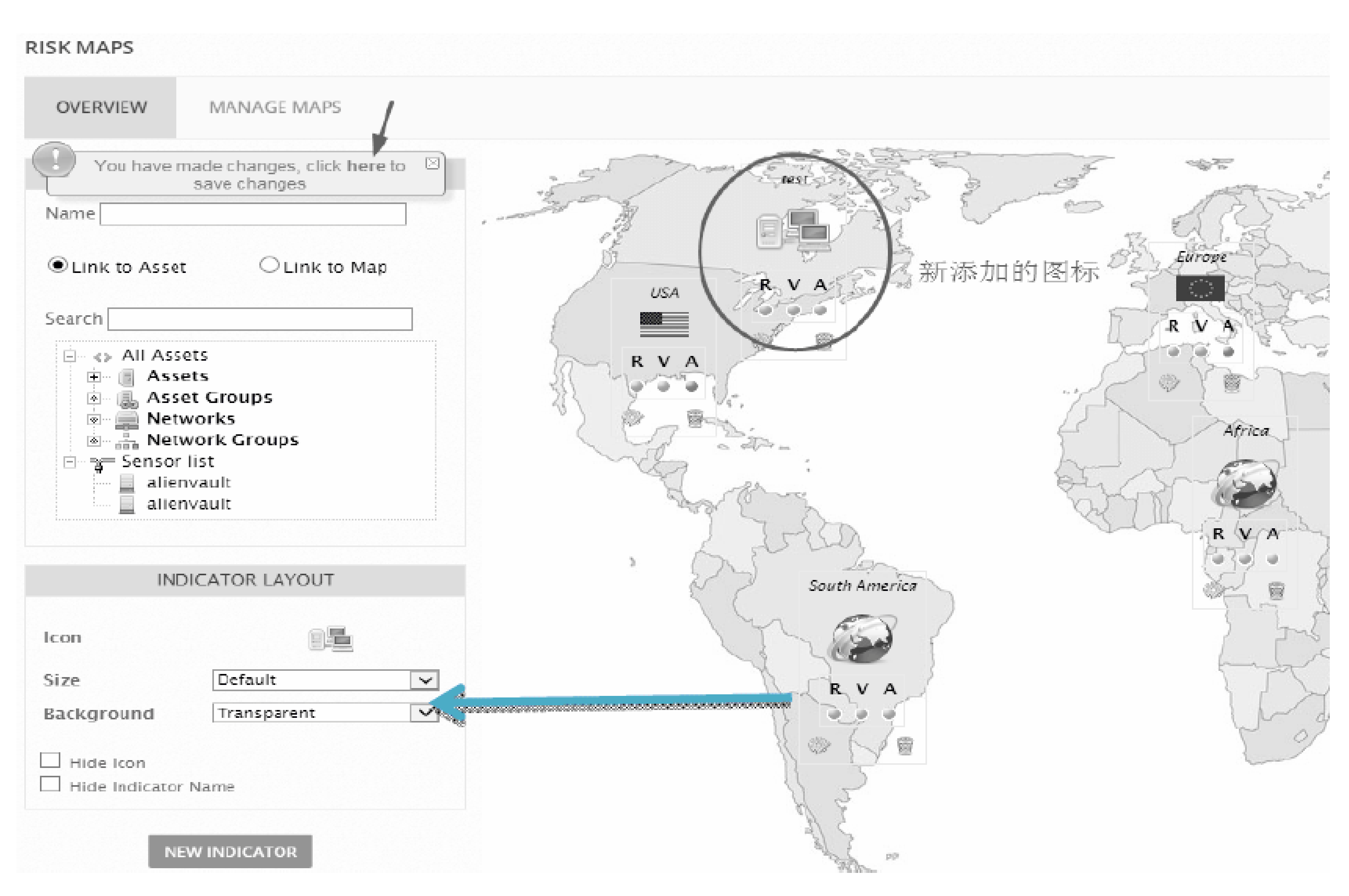


图 2-56　添加 Location 指示器

接着进入 Dashboards→Deployment status，会发现没有数据，因为这是首次设置，需要用户单击“Add location”按钮。

弹出如图 2-57 所示对话框。例如输入名称“chenguang”，在图中放大镜表单的位置输入 4 43, 91100 li, 坐标，新建立的指示器名称 test，选择需要监控的传感器名称。并单击添加传感器按钮。当我们再次单击“Deployment status”按钮时，会发现类似如图 2-58 所示画面。

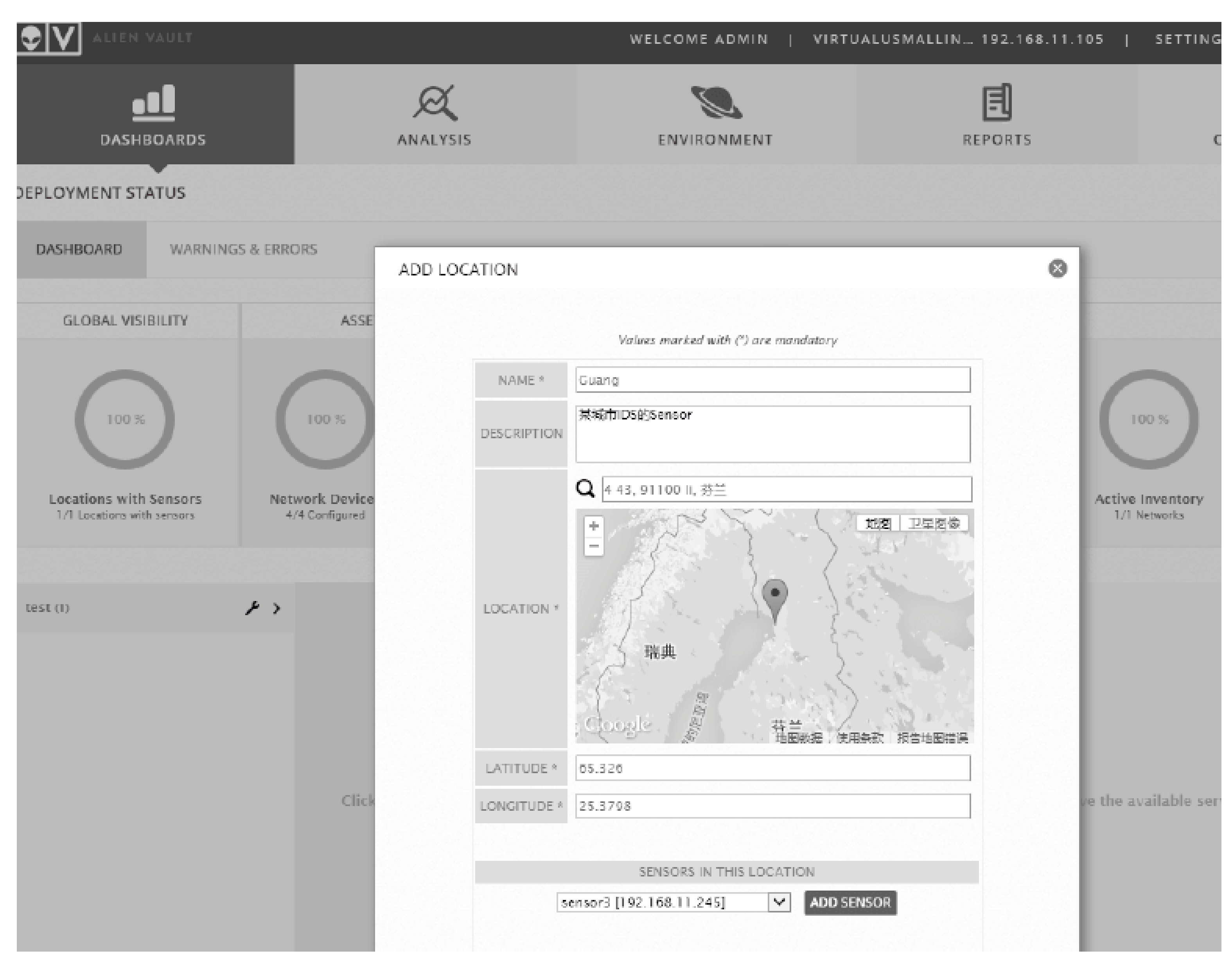


图 2-57　根据具体 Sensor 的位置添加

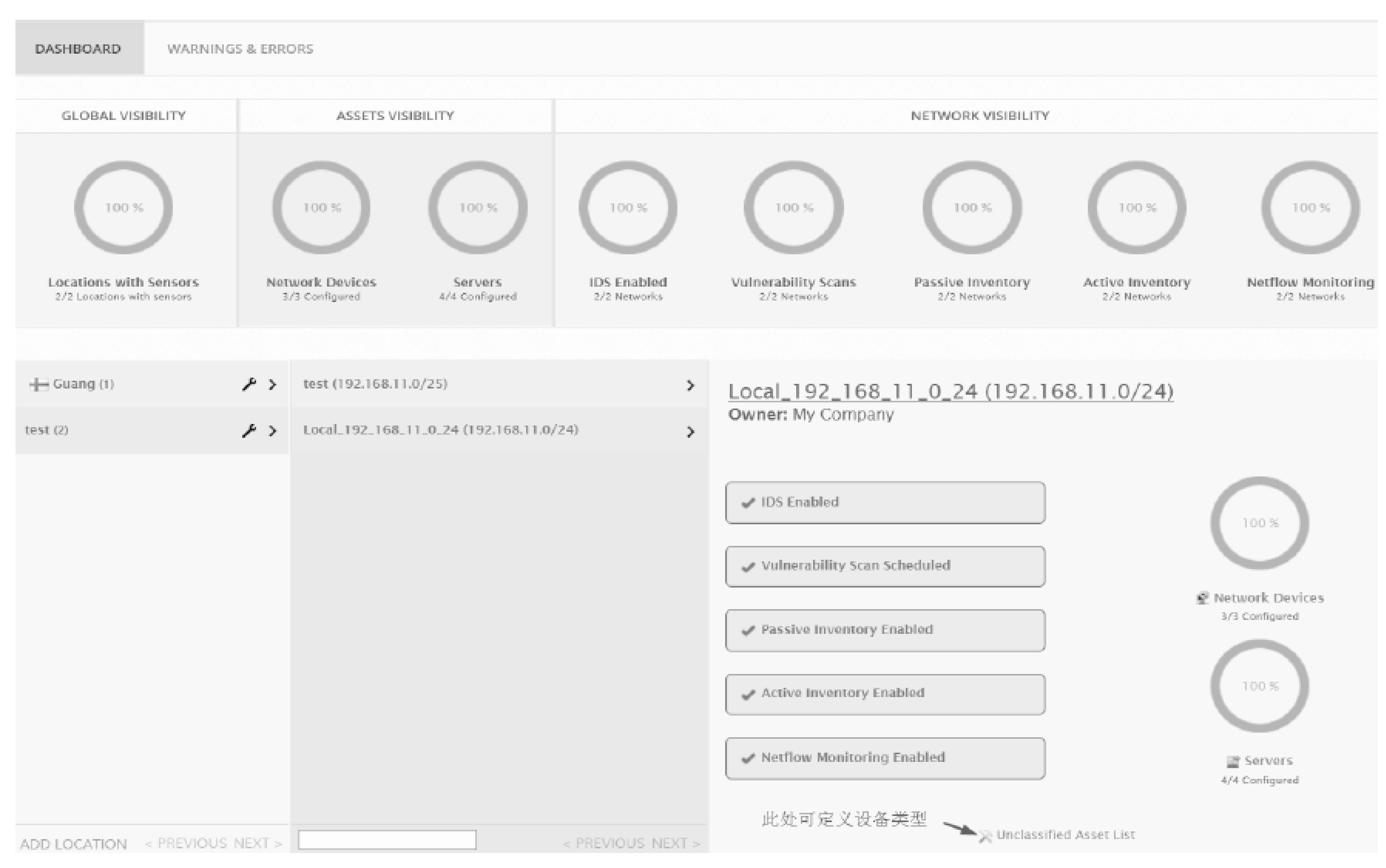


图 2-58　传感器状态展示

2.13.2　注意事项

这里需要注意，默认情况下在 Assets Visibility 下的 Server 没有选择，该图标为灰色，所以在配置主机时就必须选择设备的类型。我们可以在添加资产中定义（ADD HOST），如图 2-59 所示。

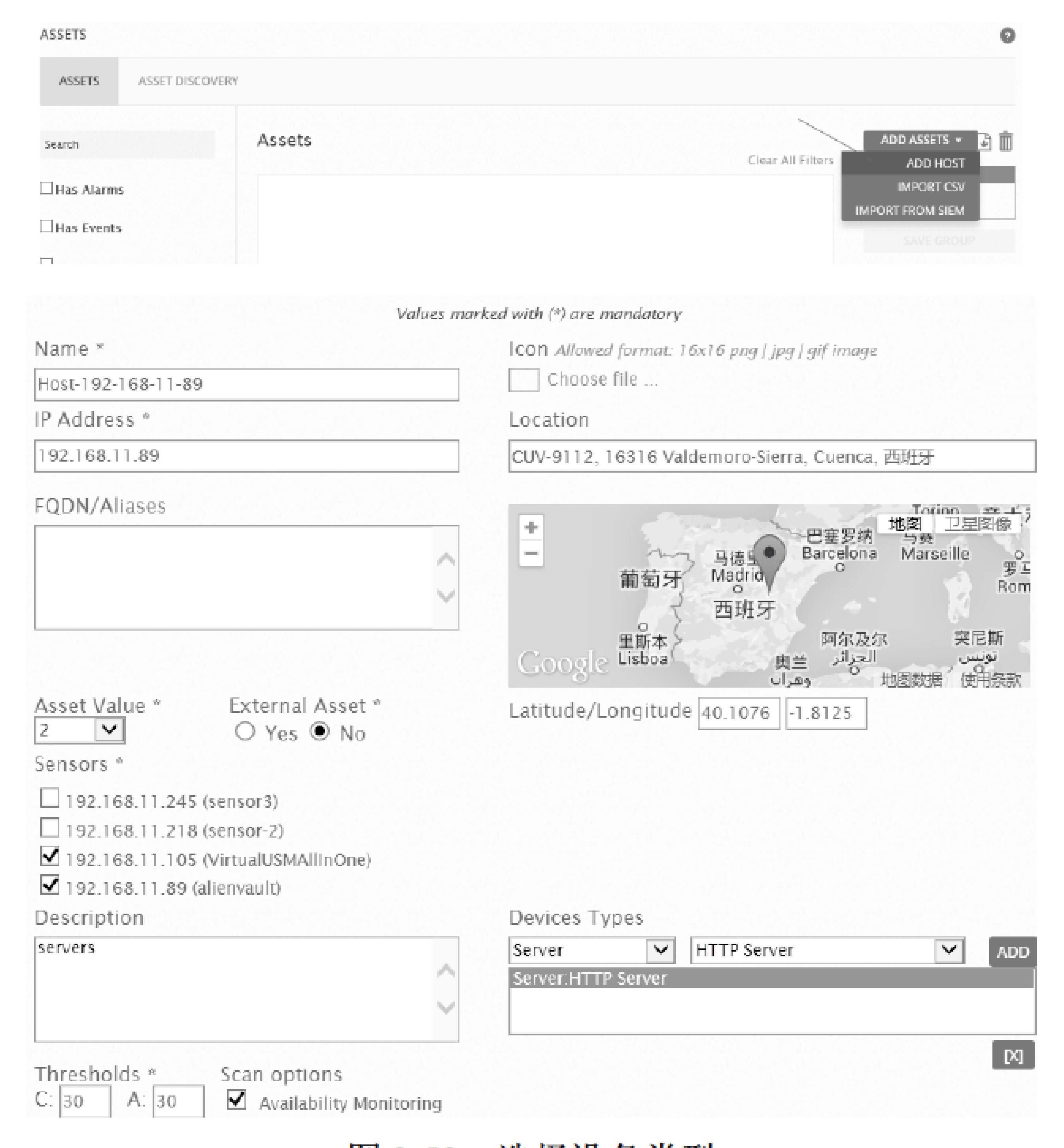


图 2-59　选择设备类型

如果在系统最初的向导设置中没有定义主机和设备类型，那么在上图这个页面上就无法显示。如果错过了最初设置向导，也可以选择“Unclassified Asset List”进行再次选择。

另外，默认情况下“Vulnerability Scan Scheduled ”没有设置，系统用红色字体标出，表示需要用户设置扫描计划。在标准的漏洞管理服务中，需要对全网中的所有资产进行周期性的漏洞扫描，并对输出结果进行格式化汇总，从而对用户提供长期的风险分析。显然单次漏洞扫描无法满足要求，所以这里提示用户设置漏洞扫描计划的频率。

2.14 安装桌面环境

默认情况下，为了提高性能 OSSIM 不提供图形环境，有些读者需要（比如数据库查询或图形化分析日志的需要）安装 X-window 环境，接下来以安装 Gnome 桌面环境举例讲解。

2.14.1 安装 GNOME 环境

Gnome 用起来方便而且安装比较简单，推荐大家使用，方法是执行以下几条命令：

```
#alienvault-update
#apt-get install gnome  (约1.22GB)
#apt-get install xserver-xorg (约15MB)
```

安装完成后还需调整 gdm 配置。

2.14.2 安装 FVWM 环境

由于 FVWM 占用内存少，启动速度快等特点，深受不少用户喜爱，其安装方法如下：

```
#apt-get install x-window-system-core fvwm
```

执行完这条命令，紧接着开始下载、解包、安装 X-Window 软件集，如果是从 OSSIM 4.3 环境下安装，下载容量约为 1.22GB，需要确保有足够的解压包和安装包的空间。下载时间就要看您的网络带宽大小，下载的文件放在/var/cache/apt/archives 目录，接着系统会解包安装，然后重启系统，最后出现图形化登录窗口。这时会发现无论普通用户，还是 root 用户都无法登录 Gnome 图形窗口，我们还需要做如下修改，重启系统并进入到单用户模式。如图 2-60 所示。

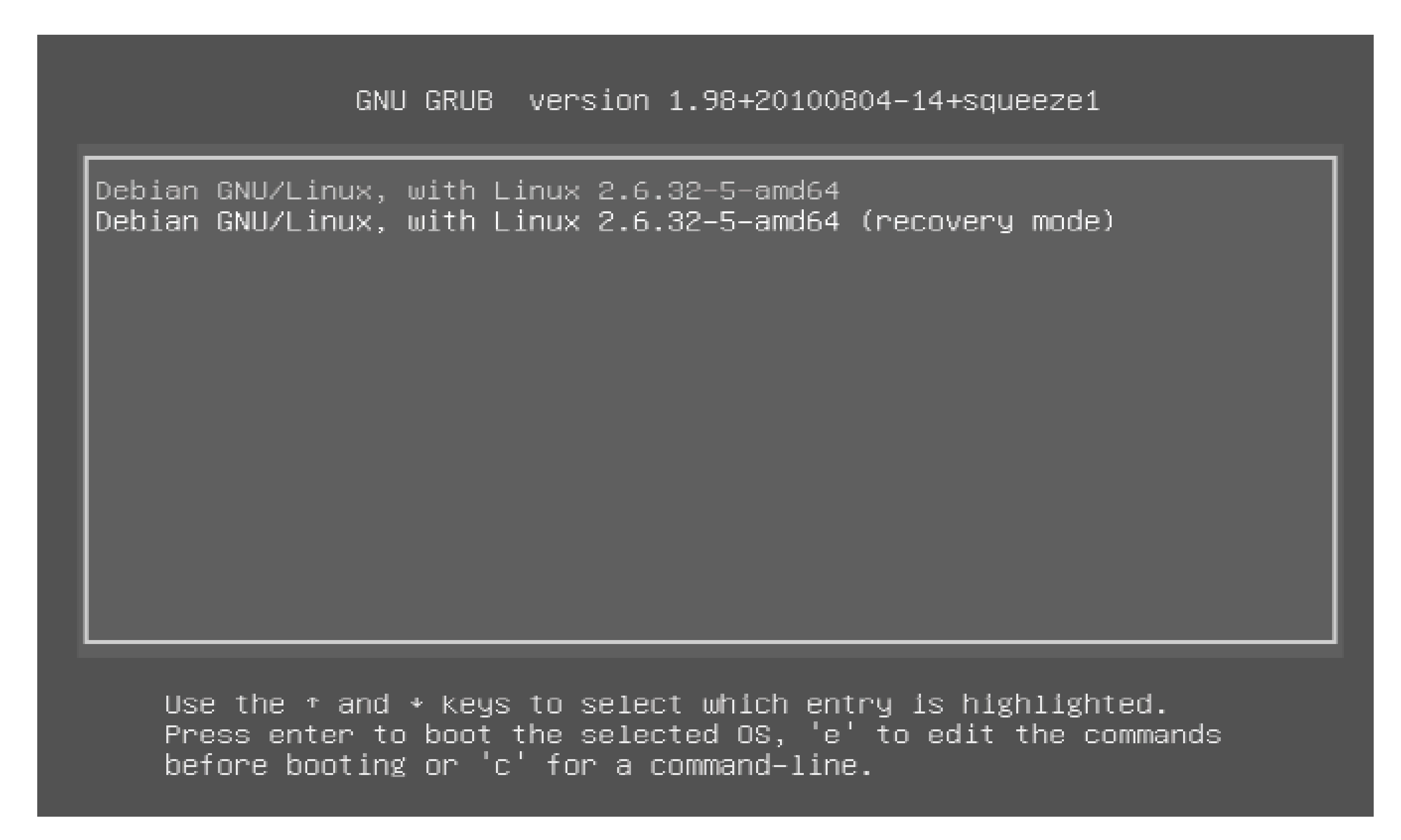


图 2-60　登录界面

新版 OSSIM 4.8 无法进入单用户模式，如何处理？首先需要赋予 grub.cfg 写权限，然后按下面方法修改：

```
#vi  /boot/grub/grub.cfg
```

将 52 行“set timeout=0”后面的数字 0，设定为较短时间，比如 5 秒。

解决方法是在 GRUB 启动界面选择 Debian GNU/Linux,with Linux 2.6.32-5-amd64（recovery mode），回车开始进入单用户模式，如图 2-61 所示。

```
(or type control-D to continue):输入口令
```

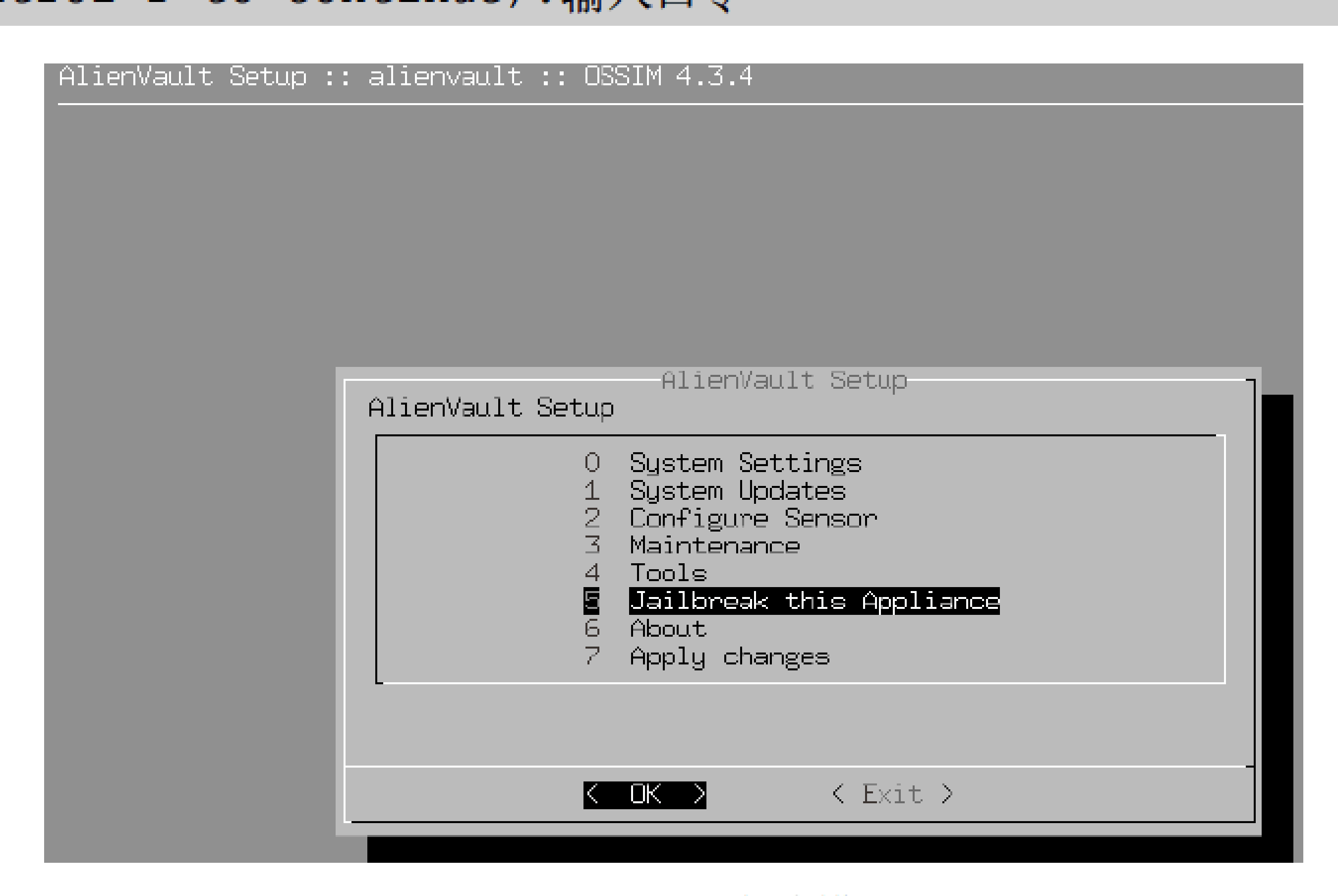


图 2-61　退出终端模式

选择图中第 5 项，进入“#”提示符。进入单用户模式后，开始修改 gdm3 下的配置文件。下面步骤以 OSSIM 4.3 系统为例进行讲解。

启动 OSSIM 系统，如果不加任何设置，以 root 登录系统会提示验证失败（因为使用了 Pam 认证机制，Pam 配置默认限制了 root 账号登录），如图 2-62 所示。只能以系统普通用户登录，下面修改配置文件可以实现 root 直接登录。

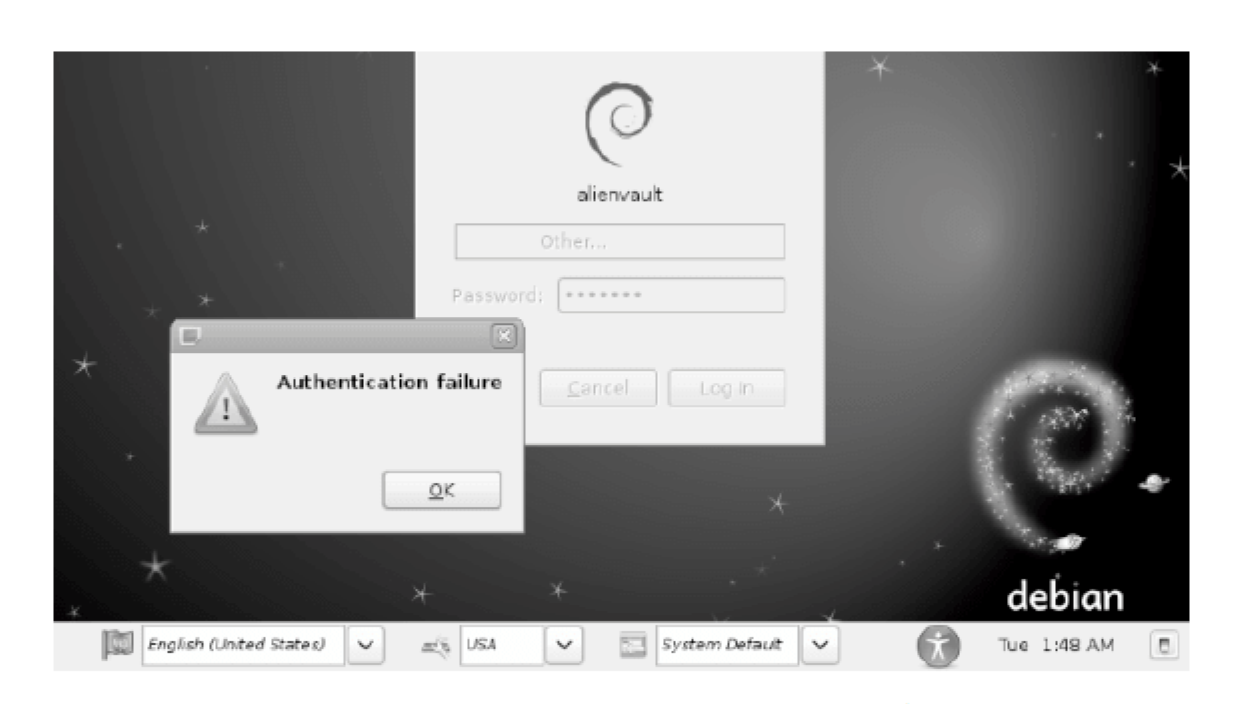


图 2-62 登录 X-Window 失败

（1）修改 gdm3 配置文件。

```
#vi /etc/pam.d/gdm3
```

注销下面这行：

```
auth required pam_succeed_if.so user != root quiet_success
```

经过以上步操作，即允许 root 用户登录图形界面。此方法适用 Debian 6/7 系统。

```
#init 2
#service gdm3 restart
```

如果图形界面出现假死，如何处理呢？可以通过以下方法解决。

首先 kill 掉 gdm3 进程，然后再启动。

```
#service gdm3 start
```

如果在 OSSIM 2.3 系统中，由于使用的是 gdm，所以只要修改/etc/gdm/gdm.conf 在 security 下加入 allowroot=true，表示允许 root 登录。

但是 OSSIM 4.x 系统使用了 gdm3，那么修改方式发生了变化。

（2）删除下载文件。

最后我们成功进入 gnome 桌面系统，记得删除下载的安装 deb 文件。

```
#rm /var/cache/apt/archives/*.deb
```

如果习惯使用 Red Hat Linux，将图形登录方式转换为字符登录的方法是将/etc/inittab 文件中的“id:5:initdefault”中的 5 换成 3 即可。但对于 OSSIM 系统（它是基于 Debian Linux）就不那么简单，它默认启动级别是 2，图形启动级别也是 2。

假设接下来我们将 OSSIM 安装在 Vware Workstations 虚拟机环境，为了方便使用图形化环境，需要安装 Vmware-Tools。如果遇到 OSSIM 系统默认没有 GCC 环境，这时如果直接安装 Vmware Tools 将会出现以下提示（如图 2-63 所示）：

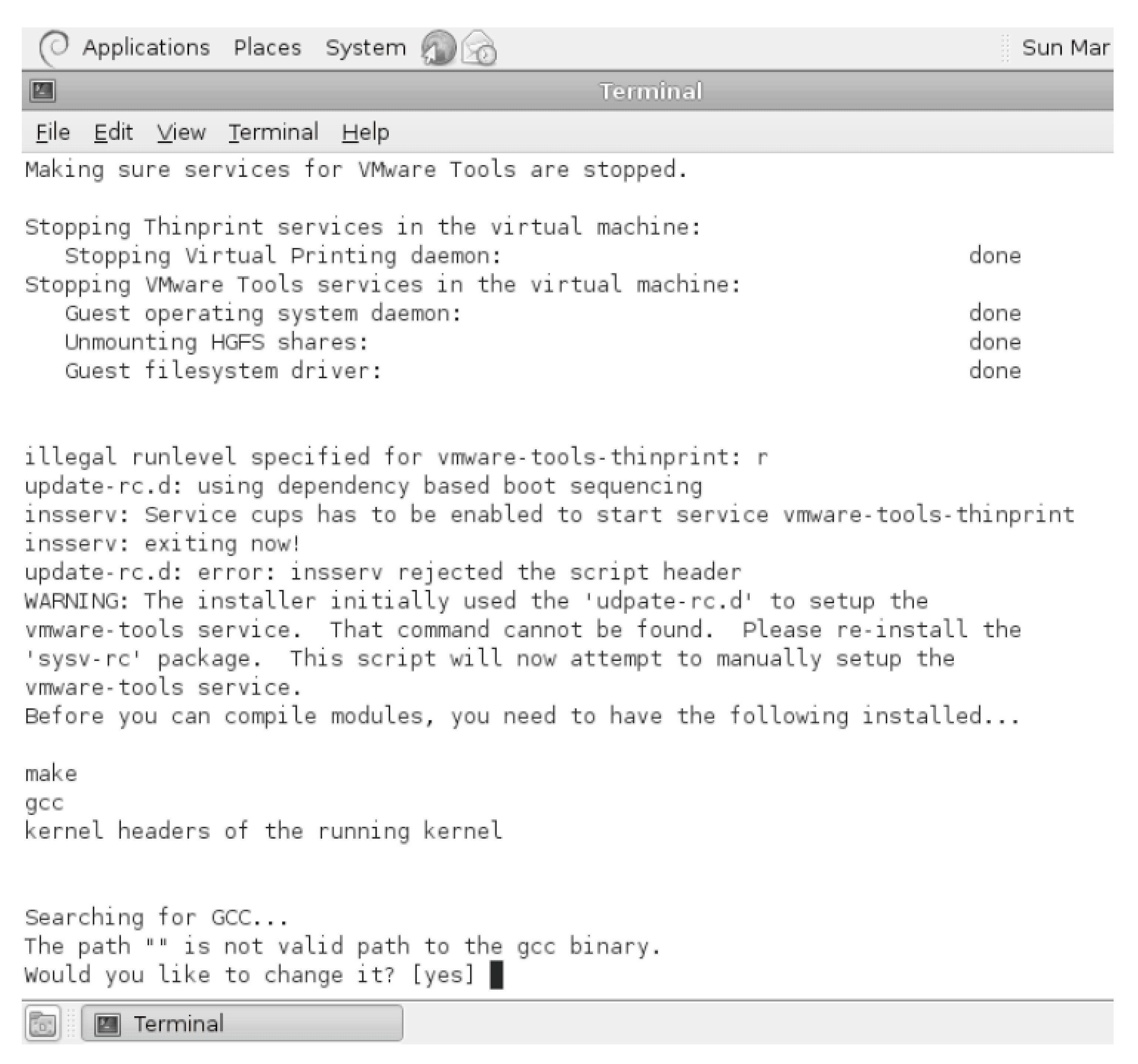


图 2-63　安装 Vmware Tools 失败提示

这表明系统缺少编译工具，chkconfig 命令是 Red Hat 公司所开发的程序，它可查询操作系统在每一个执行等级中会执行哪些系统服务，其中包括各类常驻服务。在 OSSIM 系统中除了有这款工具以外，还有 Debian 专用的 update-rc.d 工具，它也和 chkconfig 工具类似，不同的是它只是一个脚本而不是二进制程序。

当你在 Debian Linux 下安装新的应用服务器时，例如 Apache2，装完之后默认它会启动，并在下次重启后自动运行，但如果不需要自动启动也可以禁用。要实现这个目的，有个方法是手工修改/etc/rcx.d 目录的 apache2 的符号连接文件，但是效率较低，所以建议使用 update-rc.d 命令实现。

- 删除一个服务

```
#update-rc.d -f apache2 remove
```

- 增加一个服务

```
#update-rc.d apache defaults
```

在 OSSIM 系统中，另外一个管理和控制服务的工具是 invoke-rc.d，它和 Red Hat Linux 下的 service 和 ntsysv 工具很类似，更多使用方法大家可以用 man 帮助查询。在 OSSIM 下通过安装 sysv-rc-conf 工具，可以调整命令行登录和图形化登录方式。

```
#apt-get install sysv-rc-conf    \\安装工具
#sysv-rc-conf
```

如图 2-64 所示，在显示界面发现原来 Debian 默认 runlevel 2、3、4 和 5 级都是图形界面，我们现在选 runlevel 为 3，去掉 3 的 gdm。

```
File  Edit  View  Terminal  Tabs  Help
Terminal                                  Terminal

SysV Runlevel Config   -: stop service  =/+: start service  h: help  q: quit

service      1      2      3      4      5      0      6      S
----------------------------------------------------------------
fuse        [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
gdm3        [ ]    [ ]    [ ]    [x]    [x]    [ ]    [ ]    [ ]
halt        [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]
hdparm      [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
heartbeat   [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]
ifrename    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
ifupdown    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
ifupdown-$  [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
ipmievd     [ ]    [x]    [x]    [x]    [x]    [ ]    [ ]    [ ]
irqbalance  [ ]    [x]    [x]    [x]    [x]    [ ]    [ ]    [ ]
kbd         [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [ ]    [x]
```

图 2-64　配置启动服务

2.14.3　安装虚拟机

如果在虚拟机中安装 OSSIM，同时安装了 X-window，默认的分辨率只有 800×600，为了提高分辨率，还需要安装虚拟机扩展工具，下面分别针对两款常用虚拟机进行讲解。

对于在 VMware 下安装 OSSIM，为了扩展需要装上 Vmware-tools，需用到 apt-get 和 apt-cache、uname 这 3 个命令。

下面以 OSSIM 4.8 系统为例。

（1）安装 GCC（在 OSSIM 5.X 之后的版本已安装 GCC）

```
#apt-get install gcc
```

```
alienvault:~# apt-get install gcc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  gcc-4.4
Suggested packages:
  gcc-multilib manpages-dev autoconf automake1.9 libtool flex bison gdb
  gcc-doc gcc-4.4-multilib libmudflap0-4.4-dev gcc-4.4-doc gcc-4.4-locales
  libgcc1-dbg libgomp1-dbg libmudflap0-dbg libcloog-ppl0 libppl-c2 libppl7
The following NEW packages will be installed:
  gcc gcc-4.4
0 upgraded, 2 newly installed, 0 to remove and 2 not upgraded.
2 not fully installed or removed.
Need to get 2693 kB of archives.
After this operation, 4567 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://data.alienvault.com/mirror/squeeze/ squeeze/main gcc-4.4 amd64 4.4.
5-8 [2688 kB]
0% [1 gcc-4.4 2491 B/2688 kB 0%]
```

（2）安装 header-dev

再查看系统版本：

```
#uname -r
2.6.32-5-amd64
```

查看内核头文件位置：

```
#apt-cache search headers 2.6.32-5-amd64
linux-headers-2.6.32-5-amd64 - Header files for Linux 2.6.32-5-amd64
```

开始安装 linux-headers：

```
#apt-get install  linux-headers-2.6.32-5-amd64
```

```
Linux alienvault 2.6.32-5-amd64 #1 SMP Fri May 10 08:43:19 UTC 2013 x86_64 GNU/L
inux
alienvault:~# uname -r
2.6.32-5-amd64
alienvault:~# apt-cache search headers 2.6.32-5-amd64
linux-headers-2.6.32-5-amd64 - Header files for Linux 2.6.32-5-amd64
alienvault:~# apt-get install linux-headers-2.6.32-5-amd64
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  cpp-4.3 gcc-4.3 gcc-4.3-base linux-headers-2.6.32-5-common
  linux-kbuild-2.6.32
Suggested packages:
  gcc-4.3-locales gcc-4.3-multilib libmudflap0-4.3-dev gcc-4.3-doc libgcc1-dbg
  libgomp1-dbg libmudflap0-dbg
The following NEW packages will be installed:
  cpp-4.3 gcc-4.3 gcc-4.3-base linux-headers-2.6.32-5-amd64
  linux-headers-2.6.32-5-common linux-kbuild-2.6.32
0 upgraded, 6 newly installed, 0 to remove and 2 not upgraded.
2 not fully installed or removed.
Need to get 10.7 MB of archives.
After this operation, 35.3 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

以上准备工作做完，接着可以正常安装上 VMware Tools 工具。

```
   VM communication interface:                                         done
   VM communication interface socket family:                           done
   Guest filesystem driver:                                            done
   Mounting HGFS shares:                                              failed
   Blocking file system:                                               done
   Guest operating system daemon:                                      done
   Virtual Printing daemon:                                            done
The configuration of VMware Tools 8.8.0 build-471268 for Linux for this running
kernel completed successfully.

You must restart your X session before any mouse or graphics changes take
effect.

You can now run VMware Tools by invoking "/usr/bin/vmware-toolbox-cmd" from the
command line or by invoking "/usr/bin/vmware-toolbox" from the command line
during an X server session.

To enable advanced X features (e.g., guest resolution fit, drag and drop, and
file and text copy/paste), you will need to do one (or more) of the following:
1. Manually start /usr/bin/vmware-user
2. Log out and log back into your desktop session; and,
3. Restart your X session.

Enjoy,

--the VMware team
```

成功安装 VMwareTools 后对于调试使用 OSSIM 非常有利。

注意，如果选用了 Oracle VirtualBox 虚拟机安装 OSSIM，则应在 VirtualBox 程序中 Devices 菜

单下选择“InsertGuestAdditions CD image”。这时在虚拟机中的图形界面会打开 VBOXADDITIONS 虚拟光盘，我们将所有目录和文件复制到/tmp 后，然后运行./VBoxLinuxAdditions.run，并根据提示安装即可，这样即可在 OSSIM 桌面环境下支持高分辨率。

2.15 自动化配置管理工具 Ansible

从事 Hadoop 集群安装时，需要在每个节点上配置 SSH，以实现节点之间的无密码接入。下面要介绍的一款基于 Python 开源工具 Ansible，它也是和 SSH 连接密切相关，它利用推送方式对客户系统加以配置，这样所有工作都可在服务器端完成。利用 SSH 互信，这样就无须安装客户端，特别适用于分布式系统的管理，可以使用 Web UI 实现授权管理与配置，下面我们先看看它的优点：

- 无须部署代理，基于 SSH Key 方式认证，不需要在被管控主机上安装任何客户端软件（包括插件），也不需要配置数据库。
- 无须服务器，不用服务器直接使用命令。
- 使用 Python 编写，维护简单，而且基于模块化工作机制，便于扩展。

在 OSSIM 中集成 Ansible 软件包名称为 alienvault-api-core，这是 OSSIM 核心安装包之一。

2.15.1 SSH 的核心作用

借助 SSH 我们可以在远程主机上执行命令并读取输出。SSH 使用用户名和密码进行认证。在 SSH 命令的执行过程中提示输入密码。但是在自动化脚本中，SSH 命令可能在一个循环中执行多次，每次都提供密码的话，显然不实际。因此，我们需要将登录过程自动化。解决方式是可以用 SSH 密钥实现自动登录。

当中心控制服务器与远程机器通信时，Ansible 默认使用 SSH Keys 方式通信，例如，在 OSSIM Web UI 中，我们在 Alienvault Center 可查看到 Sensor 的各项性能参数。如图 2-65 所示。

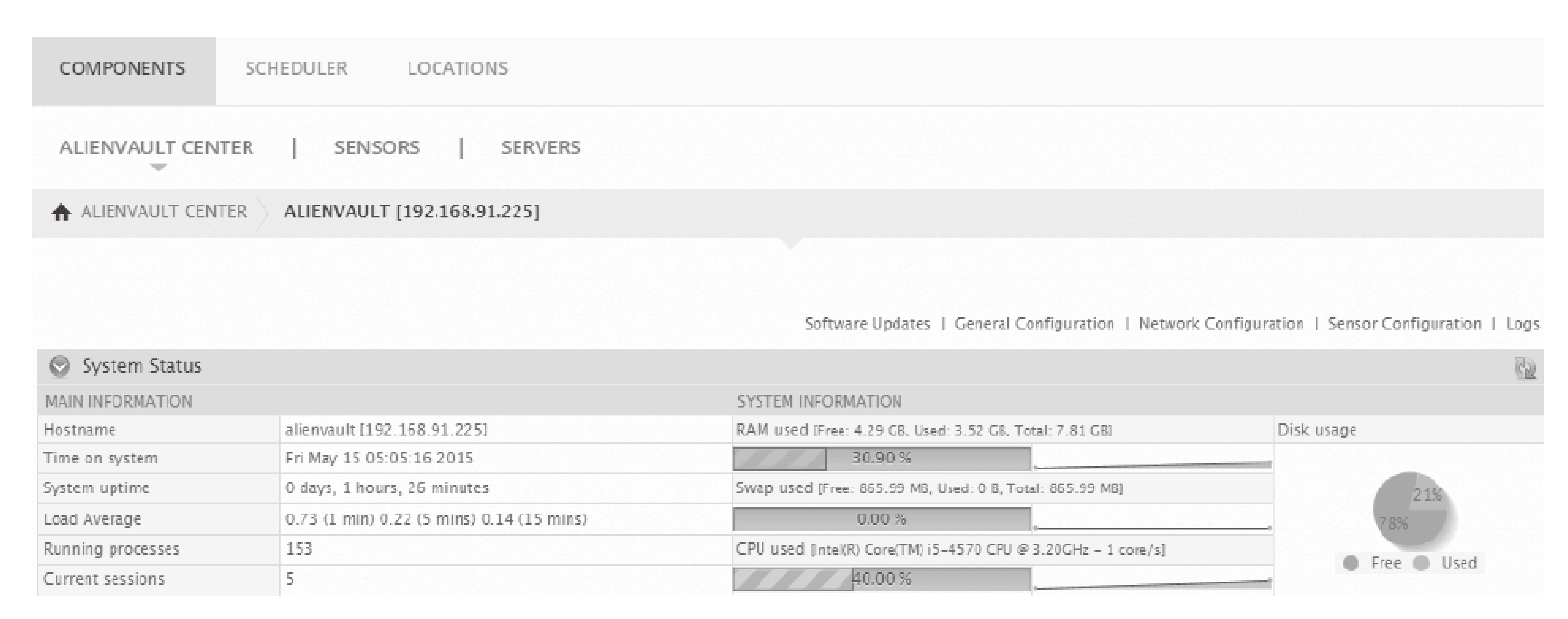


图 2-65　查看 Sensor 状态

SSH 采用基于公钥和私钥的加密技术进行自动化认证。认证密钥包含两部分：一个公钥和一个私钥。可以通过“ssh-keygen”命令创建认证密钥。要实现自动化认证，公钥必须放置在 OSSIM 服务器中（将其加入文件/home/avapi/.ssh/authorized_keys，该文件中包含了公钥 id_rsa.pub 的内容）。以 avapi 用户直接访问远程主机 192.168.91.225，其过程如图 2-66 所示。如果想了解通过 SSH 协议连接远程主机的详细过程，可输入“ssh -vvv 192.168.91.225”。

```
alienvault:/etc/ansible# ssh -v -i /var/ossim/ssl/local/private/cakey.pem  avapi
@192.168.91.225
OpenSSH_5.5p1 Debian-6+squeeze5, OpenSSL 0.9.8o 01 Jun 2010
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.91.225 [192.168.91.225] port 22.
debug1: Connection established.
debug1: permanently_set_uid: 0/0
debug1: identity file /var/ossim/ssl/local/private/cakey.pem type -1
debug1: identity file /var/ossim/ssl/local/private/cakey.pem-cert type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.5p1
debug1: match: OpenSSH_5.5p1 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze5
debug1: SSH2_MSG_KEXINIT sent
```

图 2-66　以 avapi 用户直接访问远程主机

接着，执行以下命令（结果如图 2-67 所示）：

```
# /usr/share/alienvault/api_core/bin/ansible-playbook -e
"remote_system_ip=10.32.14.10 local_system_id=<System ID>"
/etc/ansible/playbooks/auth/set_crypto_files.yml -vvvv
```

<System ID> 是服务器的 UUID，通过执行“alienvault-api about”命令获取。

“vvvv”代表输出调试信息。

```
alienvault:/etc/ansible# /usr/share/alienvault/api_core/bin/ansible-playbook -e
"remote_system_ip=192.168.91.225 local_system_id=564df90b-6ca3-f88b-7c2b-91ee2d6
42142"  /etc/ansible/playbooks/auth/set_crypto_files.yml  -vvvv

PLAY [set crypto files] *******************************************************

GATHERING FACTS ***************************************************************
<192.168.91.225> ESTABLISH CONNECTION FOR USER: avapi
<192.168.91.225> EXEC ['ssh', '-tt', '-vvv', '-o', 'StrictHostKeyChecking=no', '
-o', 'StrictHostKeyChecking=no', '-o', 'Port=22', '-o', 'IdentityFile=/var/ossim
/ssl/local/private/cakey_avapi.pem', '-o', 'KbdInteractiveAuthentication=no', '-
o', 'PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey',
 '-o', 'PasswordAuthentication=no', '-o', 'User=avapi', '-o', 'ConnectTimeout=10
', '192.168.91.225', "/bin/sh -c 'mkdir -p $HOME/.ansible/tmp/ansible-1432025867
.73-156347916334100 && chmod a+rx $HOME/.ansible/tmp/ansible-1432025867.73-15634
7916334100 && echo $HOME/.ansible/tmp/ansible-1432025867.73-156347916334100'"]
<192.168.91.225> REMOTE_MODULE setup
<192.168.91.225> PUT /tmp/tmp_bUNbn TO /home/avapi/.ansible/tmp/ansible-14320258
67.73-156347916334100/setup
<192.168.91.225> EXEC ['ssh', '-tt', '-vvv', '-o', 'StrictHostKeyChecking=no', '
-o', 'StrictHostKeyChecking=no', '-o', 'Port=22', '-o', 'IdentityFile=/var/ossim
```

图 2-67　ansible-playbook 命令执行结果

该脚本，当执行到“get remote system id”这一步时很慢，大约经过几分钟才完成。执行

过程中可以按 Ctrl+C 终止。

2.15.2　Ansible 配置

在 OSSIM 系统中，ansible 配置文件位于/etc/ansible/ansible.cfg，其中 Hostfile=/etc/ansible/hosts 指定默认 hosts 配置的 IP 地址，该 IP 是分布式 OSSIM 系统中的 Sensor 主机 IP。另外，Ansible 二进制命令默认不能在命令行下执行，建议大家输入完整路径。

2.15.3　Ansible 实战

下面我们在 OSSIM 环境下，通过几个命令测试存活的 Sensor 主机。

（1）向远程主机执行第一条命令

读者对于这个命令和加载模块不理解，暂时没有关系，不影响实验。首先输入下面命令（如图 2-68 所示）：

```
# /usr/share/alienvault/api_core/bin/ansible all -m ping
```

```
alienvault:~# /usr/share/alienvault/api_core/bin/ansible all -m ping
192.168.91.223 | success >> {
    "changed": false,
    "ping": "pong"
}
```

图 2-68　向远程主机执行 ping 命令

如收到响应，则说明远程主机存活。如果想对远程机器进行管理，那么首先需要将远程机器的主机名或 IP 写入 Ansible 清单文件/etc/ansible/hosts 中，在 Ansible 中称其为清单文件，里面包含了所有要管理的机器。该文件中的内容一般是在你添加 Sensor 过程中自动完成。注意：如果启用密码验证，则使用“--ask-pass”选项，例如（如图 2-69 所示）：

```
# /usr/share/alienvault/api_core/bin/ansible all -m ping --ask-pass
```

```
alienvault:/etc/ansible# /usr/share/alienvault/api_core/bin/ansible 192.168.91.2
25 -m ping --ask-pass
SSH password:
192.168.91.225 | success >> {
    "changed": false,
    "ping": "pong"
}
```

图 2-69　向远程主机执行 ping 命令

接下来，我们需要获取远程主机运行时间、在线用户平均负载等信息，继续输入命令：

```
# /usr/share/alienvault/api_core/bin/ansible all -m raw -a 'w'
```

运行效果如图 2-70 所示。

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible all -m raw -a 'w'
192.168.91.223 | success | rc=0 >>
 01:18:26 up  3:04,  2 users,  load average: 0.01, 0.10, 0.16
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1                      22:22    0.00s  0.85s  0.02s ssh -tt -q -o S
avapi    pts/1    localhost.alienv 01:18    0.00s  0.00s  0.00s w

192.168.91.224 | success | rc=0 >>
 01:18:26 up  1:59,  2 users,  load average: 0.01, 0.02, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1                      23:21   10.00s  1.23s  0.03s /bin/bash
avapi    pts/0    192.168.91.223   01:18    0.00s  0.00s  0.00s w
```

图 2-70　获取远程主机运行时间信息

瞧，对方并没有提示输入密码。Ansible 是基于模块的应用，通过实现设定的模块来完成一些远程管理工作，Ansible 模块众多（约 180 个），要查看可用的模块，可以使用指令“ansible-doc –l”。

（2）查看远程主机基本信息

```
#/usr/share/alienvault/api_core/bin/ansible all -m setup
```

该命令输出非常丰富，读者自己可以在实验机上执行一下看看。测试远程主机运行状态如下：

```
#/usr/share/alienvault/api_core/bin/ansible all -m ping
```

（3）远程创建文件符号链接

远程创建文件符号链接命令执行结果如图 2-71 所示。

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible all -m file -a "src=/etc/resolv.conf dest=/tmp/resolv.conf state=link"
192.168.91.223 | success >> {
    "changed": false,
    "dest": "/tmp/resolv.conf",
    "gid": 103,
    "group": "alienvault",
    "mode": "0777",
    "owner": "avapi",
    "size": 16,
    "src": "/etc/resolv.conf",
    "state": "link",
    "uid": 101
}
```

图 2-71　创建符号连接

选项“src”表示被赋值到远程主机的本地文件，注意是绝对路径。如果是目录，将递归赋值。

选项“dest”表示要将源文件复制到的远程主机的绝对路径。如果源文件是目录，那么该路径也必须为目录。

（4）远程文件信息查看举例（如图 2-72 所示）

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible all -m file -a "ls
 -la /tmp/resolv.conf
"
192.168.91.223 | FAILED >> {
    "failed": true,
    "msg": "this module requires key=value arguments (['ls', '-la', '/tmp/resolv
.conf'])"
}

192.168.91.224 | FAILED >> {
    "failed": true,
    "msg": "this module requires key=value arguments (['ls', '-la', '/tmp/resolv
.conf'])"
}

alienvault:~/.ssh# _
```

图 2-72　查看远程文件

执行完这条命令，我们看看本地/tmp 目录下发生的变化，如图 2-73 所示：

```
alienvault:/tmp# ls -l
total 4
-rw-r--r-- 1 root  root        3924 Mar  3 01:25 monit.state
lrwxrwxrwx 1 avapi alienvault    16 Mar  3 01:24 resolv.conf -> /etc/resolv.conf
alienvault:/tmp#
```

图 2-73　观察符号连接文件

（5）删除远程文件符号链接

删除远程文件符号链接命令执行结果如图 2-74 所示。

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible all -m file -a "pa
th=/tmp/resolv.conf state=absent"
192.168.91.223 | success >> {
    "changed": true,
    "path": "/tmp/resolv.conf",
    "state": "absent"
}

192.168.91.224 | success >> {
    "changed": true,
    "path": "/tmp/resolv.conf",
    "state": "absent"
}

alienvault:~/.ssh# _
```

图 2-74　删除符号连接

（6）在远程主机上执行命令 uptime

远程主机上执行命令 uptime，结果如图 2-75 所示。

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible all -m raw -a "upt
ime"
192.168.91.223 | success | rc=0 >>
 01:46:00 up  3:31,  4 users,  load average: 0.08, 0.10, 0.11


192.168.91.224 | success | rc=0 >>
 01:46:01 up  2:27,  2 users,  load average: 0.33, 0.10, 0.03


alienvault:~/.ssh#
```

图 2-75　执行 uptime

将上面命令中的“raw”参数替换成 command 同样可以。

（7）复杂操作——将脚本分发到远程主机并执行

远程部署脚本可以实现批量配置和程序部署，注意的是，ansible 本身并不能批量装在多个机器上，这里指它可以通过 ansible 运行的模块提供一种框架，以实现批量部署。下面看个例子来体会一下，在整个过程中需要输入用户名和密码吗？不用每段过程都一气呵成。首先建立如下脚本 test.sh，如图 2-76 所示。

将脚本分发到远程 Sensor，如图 2-77 所示。

```
alienvault:~/.ssh# chmod +x ./test.sh
alienvault:~/.ssh# ./test.sh
2015-03-03_01:48:49
alienvault:~/.ssh# cat test.sh
#!/bin/sh
date +%F_%H:%M:%S
alienvault:~/.ssh# _
```

图 2-76　建立测试脚本

```
192.168.91.224 | FAILED >> {
    "failed": true,
    "gid": 103,
    "group": "alienvault",
    "mode": "0755",
    "msg": "chown failed",
    "owner": "avapi",
    "path": "/tmp/test.sh",
    "size": 28,
    "state": "file",
    "uid": 101
}
```

图 2-77　分发脚本

远程执行脚本，如图 2-78 所示。

```
alienvault:~/.ssh# /usr/share/alienvault/api_core/bin/ansible -m shell all -a "/
tmp/test.sh"
192.168.91.223 | success | rc=0 >>
2015-03-03_02:12:07

192.168.91.224 | success | rc=0 >>
2015-03-03_02:12:07
```

图 2-78　执行脚本

下面接着分析 SSH/Ansible 在 OSSIM 下是如何发挥作用的，我们在 OSSIM Server 下做如下实验，注意这个实验是为了理解它们之间的关系和作用，不要在上线系统进行该项实验。

（1）删除 tmp 临时目录及文件。

```
# rm -R /root/.ansible/tmp/*
# rm -R /home/avapi/.ansible/tmp/*
```

（2）产生新的 rsa。

```
# ssh-keygen -t rsa
```

默认会保存到/root/.ssh/id_rsa。

```
# ssh avapi@192.168.11.100 mkdir -p .ssh      \\注意后面使用主机名称
```

（3）为 avapi 用户创建密码。

```
#passwd avapi
```

（4）复制密钥。

```
# cat /root/.ssh/id_rsa.pub | ssh avapi@192.168.11.100 'cat
>> .ssh/authorized_keys'
```

（5）执行 ssh avapi@192.168.11.100。

（6）执行 ossim-reconfig。

（7）执行 alienvault-api add_system --system-ip=XX.XX.XX.XX --password=XXXXXX。

2.15.4　丰富的模块

其他常用模块包括 service（系统服务管理）、cron（计划任务管理）、synchronize（使用 rsync 同步文件）、user（系统用户管理）等，详细内容大家可以通过以下命令了解。

```
#/usr/share/alienvault/api_core/bin/ansible-doc  -l
```

2.15.5　Ansible 与其他配置管理的对比

笔者列举了目前几款主流的、与 Ansible 功能类似的配置管理软件 Puppet，这里所做的对比不针对各个软件的性能作比较，只是对各个软件的特性做比较。具体内容如表 2-7 所示。

表 2-7　Ansible 与 puppet 对比

比较项目	Puppet	Ansible
开发语言	Ruby	Python
客户端	有	无
服务器与远程机器是否相互验证	是	是
服务器与远程机器通信是否加密	是，标准 SSL 协议	是，使用 OpenSSH
平台支持	Unix/Linux/BSD/Windows	Unix/Linux/BSD/Windows
是否提供 Web UI	提供	商业版提供
配置文件格式	Ruby 语法格式	YAML
命令行执行	不支持，但可通过配置模块实现	支持

2.16　SIEM 控制台基础

无论 OSSIM 的版本如何发展，SIEM 控制台分析是基础。下面从 SIEM 分析方法开始讲起，SIEM 事件控制台位于 ANALYSIS→SECURITY EVENTS（SIEM）菜单，如图 2-79 所示。SIEM 控制台是基于事件数据库的搜索引擎，能够让管理人员用更加集中的方式，针对整个系统的安全状态进行分析。

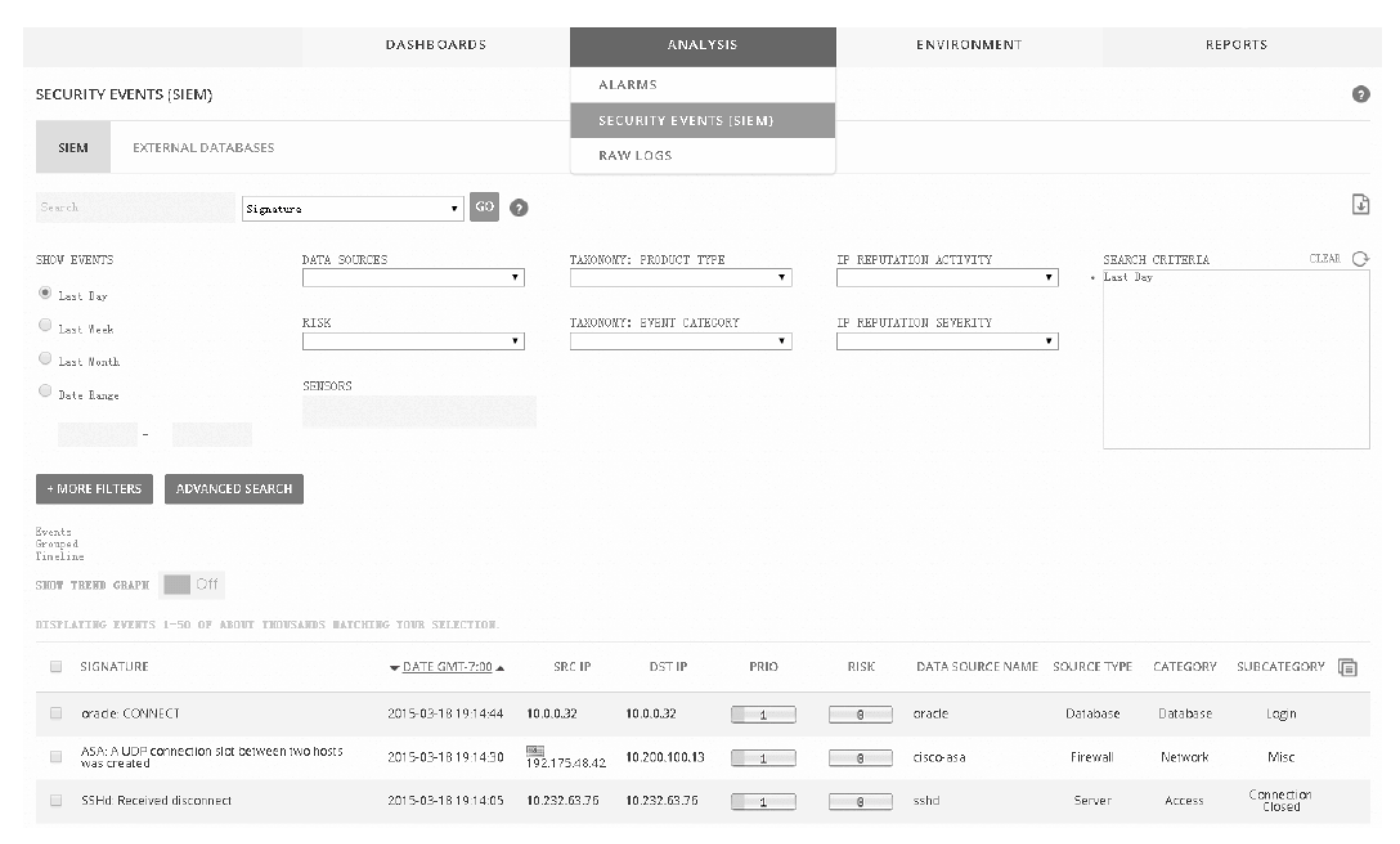


图 2-79　OSSIM 4.8 SIEM 控制面板

2.16.1　SIEM 控制台日志过滤技巧

在 OSSIM 的 SEIM 控制台中可以显示大量日志和报警，从整体数量上看占据前三甲的包括 OSSEC、Syslog 收集的各类日志以及 Snort 事件（网络数据包深度分析），其他事件的过滤可以通过选择 Data Sources 实现，如图 2-80 所示。

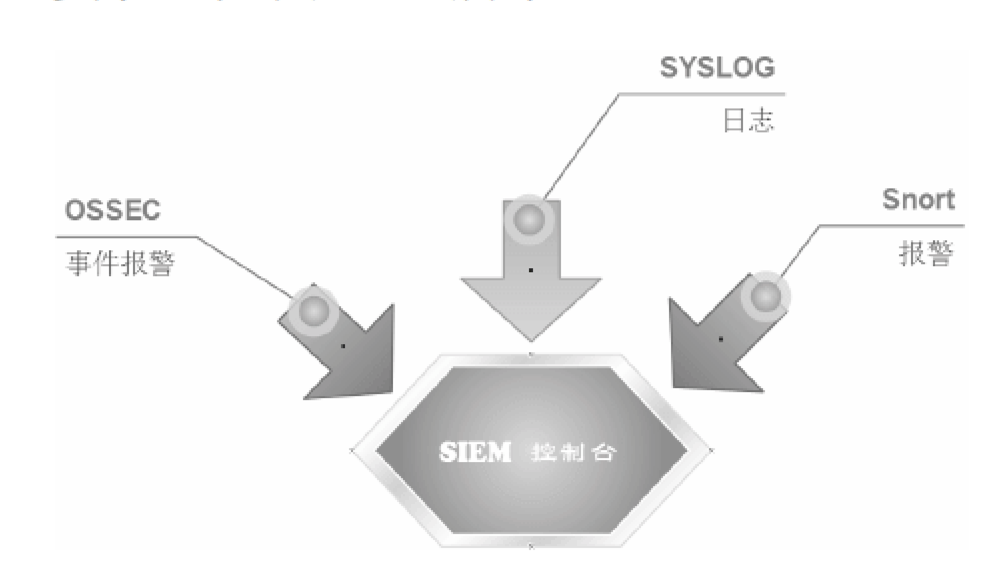


图 2-80　SIEM 事件主要构成

在事件列表最下方有三个重要参数需要大家理解：

（1）priority threshold：优先级阈值。

（2）Active eventWindows（days）：事件被存储在指定的空间，称为活动时间窗口，表示你可以在 SIEM 控制台里查询到的时间，一般是 5 天内，企业版中时间更长，即提供在线查询的保存天数，如果超过 5 天的事件将自动归档到磁盘中，也可以从归档文件恢复到数据库中，保存为 SQL 文件。

（3）Active event windows(events)：在上一条中定义了查询时间，这里定义了窗口中事件

的数量为 4M 条，也就是 4×10^6 条。通过调度任务系统，将旧的时间从数据库中写入到磁盘，当然也可以从 SQL 文件中恢复某时段的事件。

SIEM 合并冗余的报警信息主要是从 3 方面来考虑：

（1）合并基于主机的监控 Ossec 产生的冗余报警事件。

（2）合并基于网络的监控 Snort 产生的冗余报警事件。

（3）合并来自 Directive 的告警。

接着，我们查看 SIME 的数据源中的分类。如图 2-81 所示。

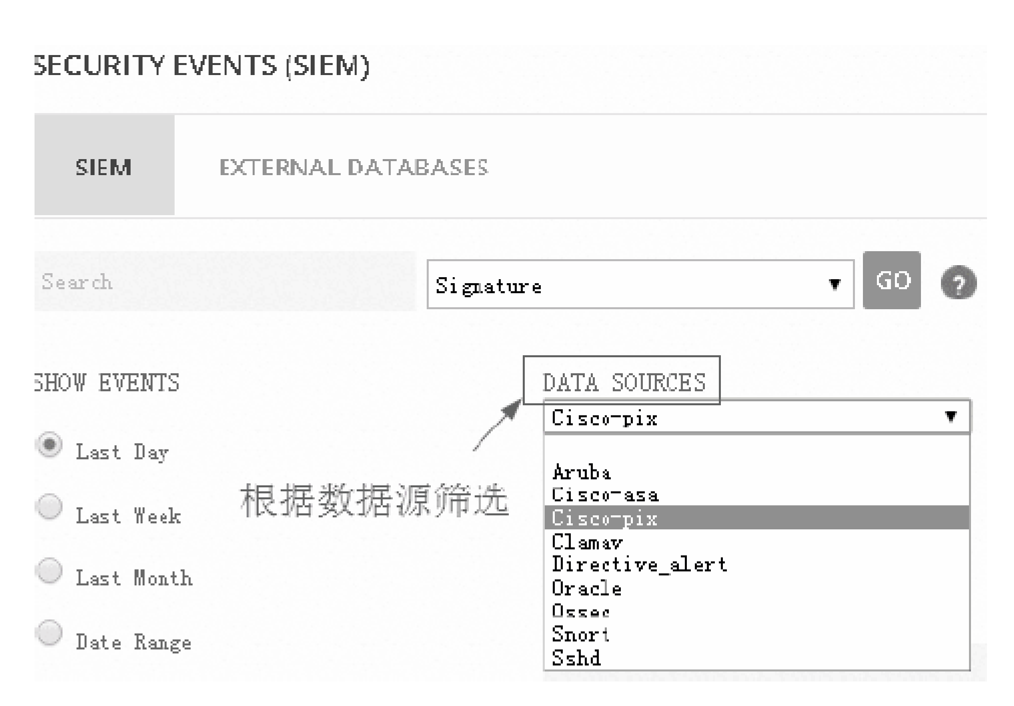


图 2-81　根据数据源筛选事件

报警日志的过滤实质是保留或者抛弃所关心的日志，将原始日志消息解析为统一格式，以便分析数据。在 OSSIM 的 SIEM 控制台中能显示很多数据，如何快速过滤出有用的数据至关重要。

首先，我们认识 SIEM 日志基本格式，它由 Signature、Date、Sensor、Source、 Destination、Asset 和 Risk 共七个部分组成，各部分含义如下：

- Signature：日志特征。
- Date：时间。
- Sensor：传感器，表示从哪个探测器获取的日志。
- Source：源地址，往往是攻击的发源地，实际上这个地址很可能伪造。
- Destination：目的地址，通常在攻击期间这个地址保持不变。
- Asset：资产。
- Risk：风险值。

但是，我们可以在 ▶ Custom Views 自定义显示方式中获得更多的日志信息，如图 2-82 所示。

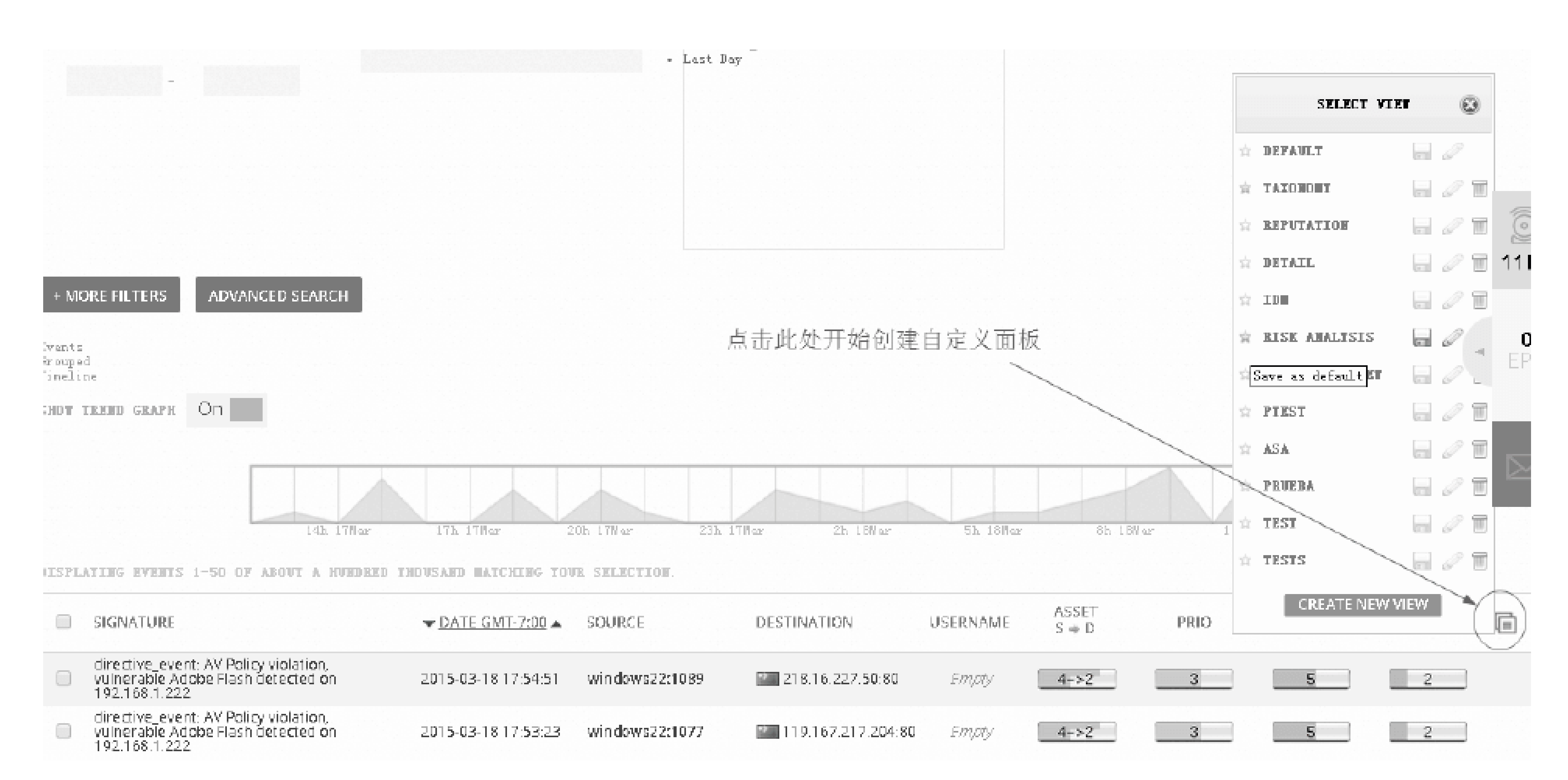


图 2-82　自定义 SIEM 列表

在 SIEM 面板中有很多过滤开关，从上至下依次介绍如下：

（1）首先是 Search，它可以输入日志的关键字，再单击“Signature”按钮，系统就会列出与之匹配的日志，然后再找出这些日志，但其中肯定还有不少干扰日志。接着进一步过滤，输入 IP 地址，然后单击“IP”按钮，它会列出“src or dst ip”、“src ip”、“dst ip”、“src or dst host”、“src host”和“dst host”6 种筛选方式。

经过多重筛选后，基本就能定位到想要的日志。如图 2-83 所示。我们还可以通过单击“clear”按钮来逐条删除过滤条件。

图 2-83　列出当前查询条件

（2）其次，可根据“Sensor+数据源”组合过滤模式，我们可以输入探测器 IP 地址，然后输入数据源种类以及风险等级的低、中、高来更精确地过滤日志，在提供的更多过滤选项中还可以由数据源组、网络/主机组以及日志种类等特性进行过滤。如图 2-84 所示。

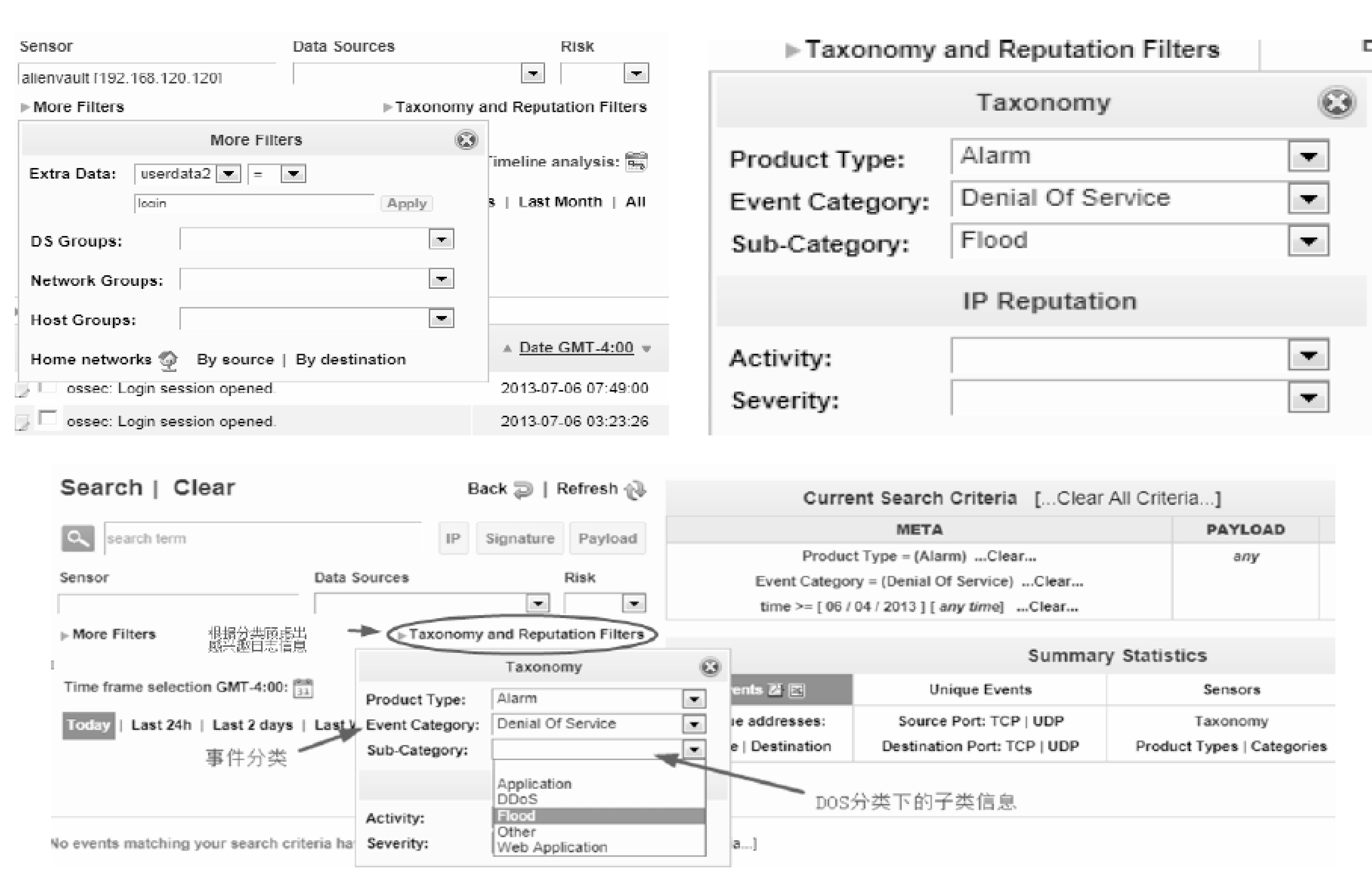


图 2-84　SIEM 过滤选项

另外，SIEM 提供的日志统计功能非常实用，它可以按不同的地址、不同的日志（例如根据协议和端口来分类）、不同的嗅探器和不同的数据源对收集来的日志进行分类统计，如图 2-85 所示。

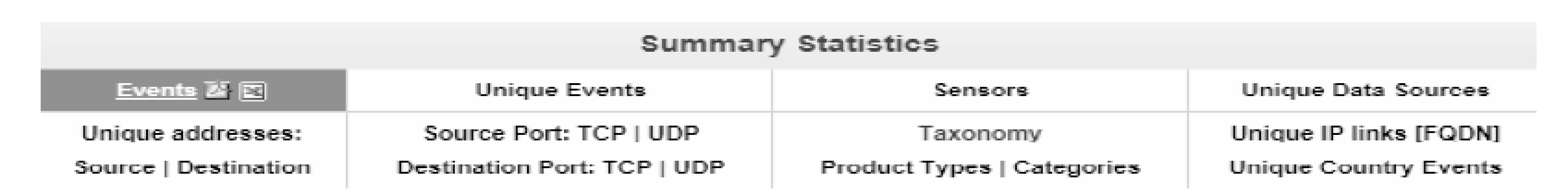


图 2-85　分类统计

在 SIEM 事件列表中往往显示多种类型，如果想要过滤某一种数据源日志，则使用过滤源的方式。如图 2-86 所示。

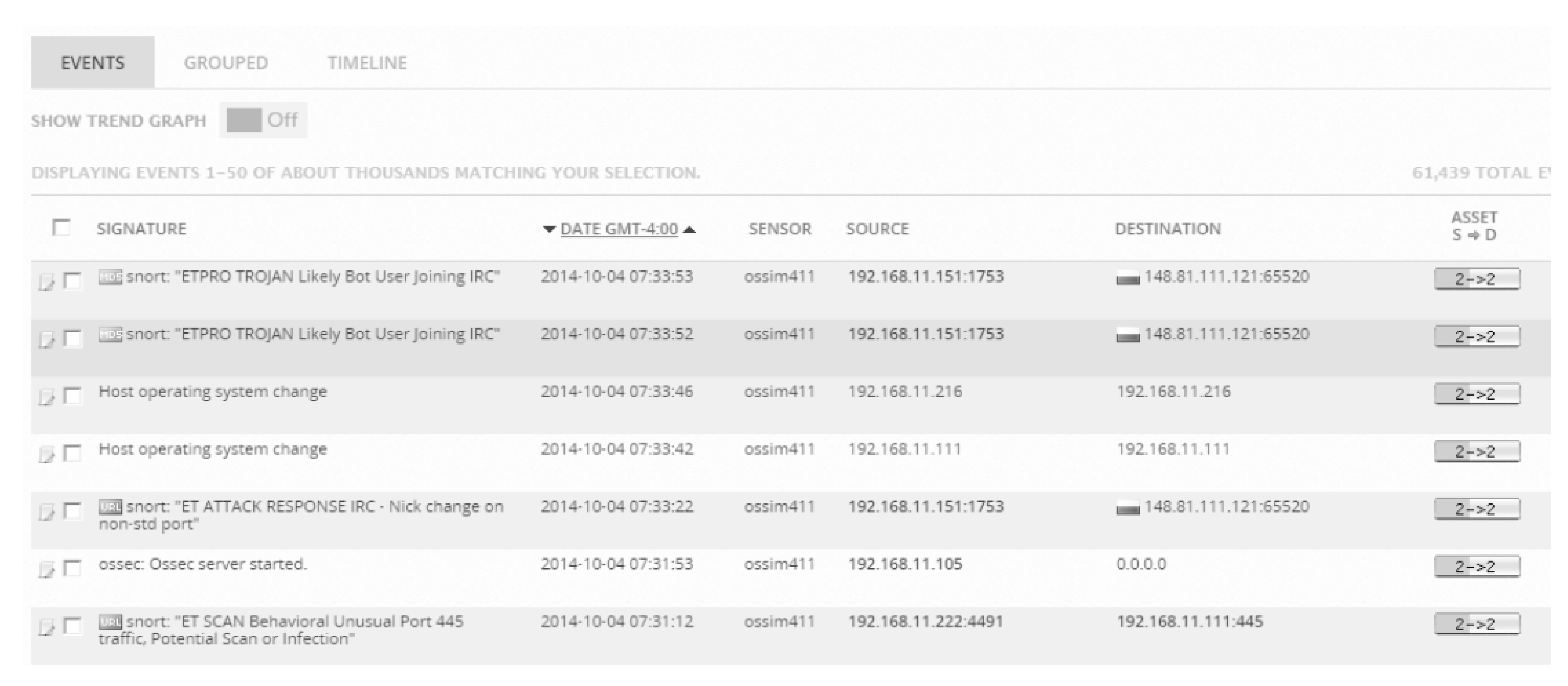


图 2-86　SIEM 事件过滤

我们可以选择数据源过滤，例如选择 Snort，如图 2-87 所示。在 SIEM 日志分析中，有时

希望能实时显示 OSSIM Server 端处理的事件，我们可以选择图 2-87 中“REAL-TIME”按钮，在显示界面中会不停地刷新事件，我们还能通过“Show plugin filter”功能过滤出指定插件发送来的事件。

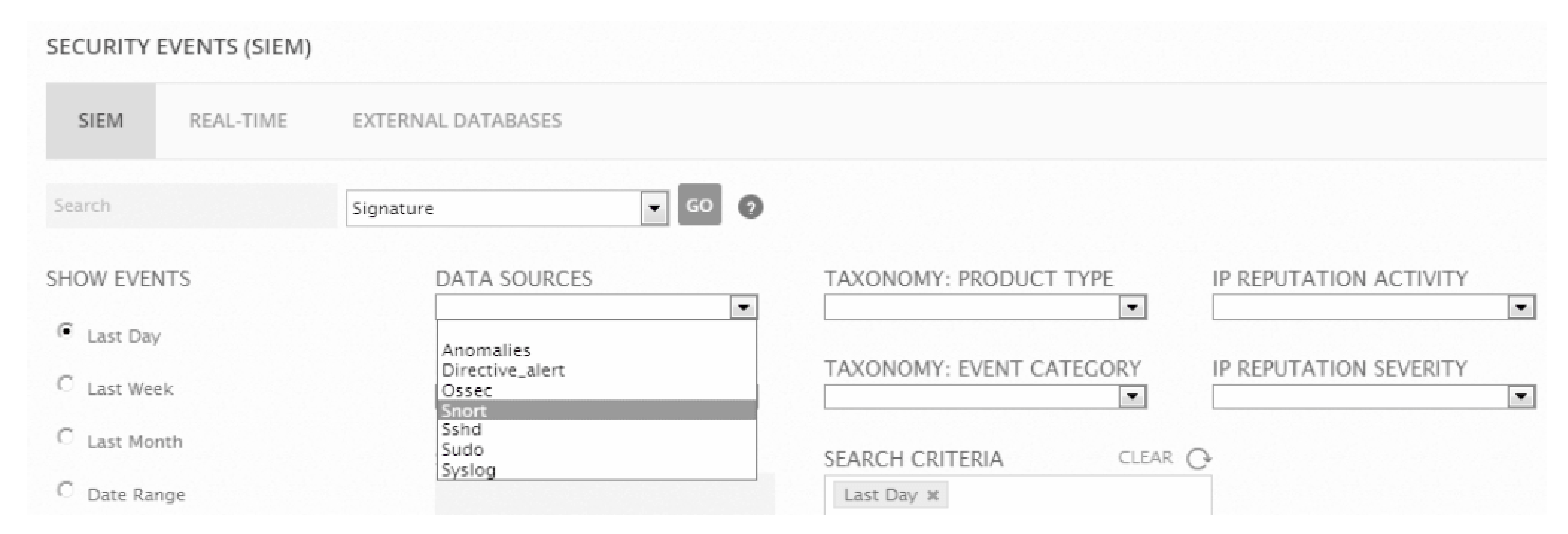
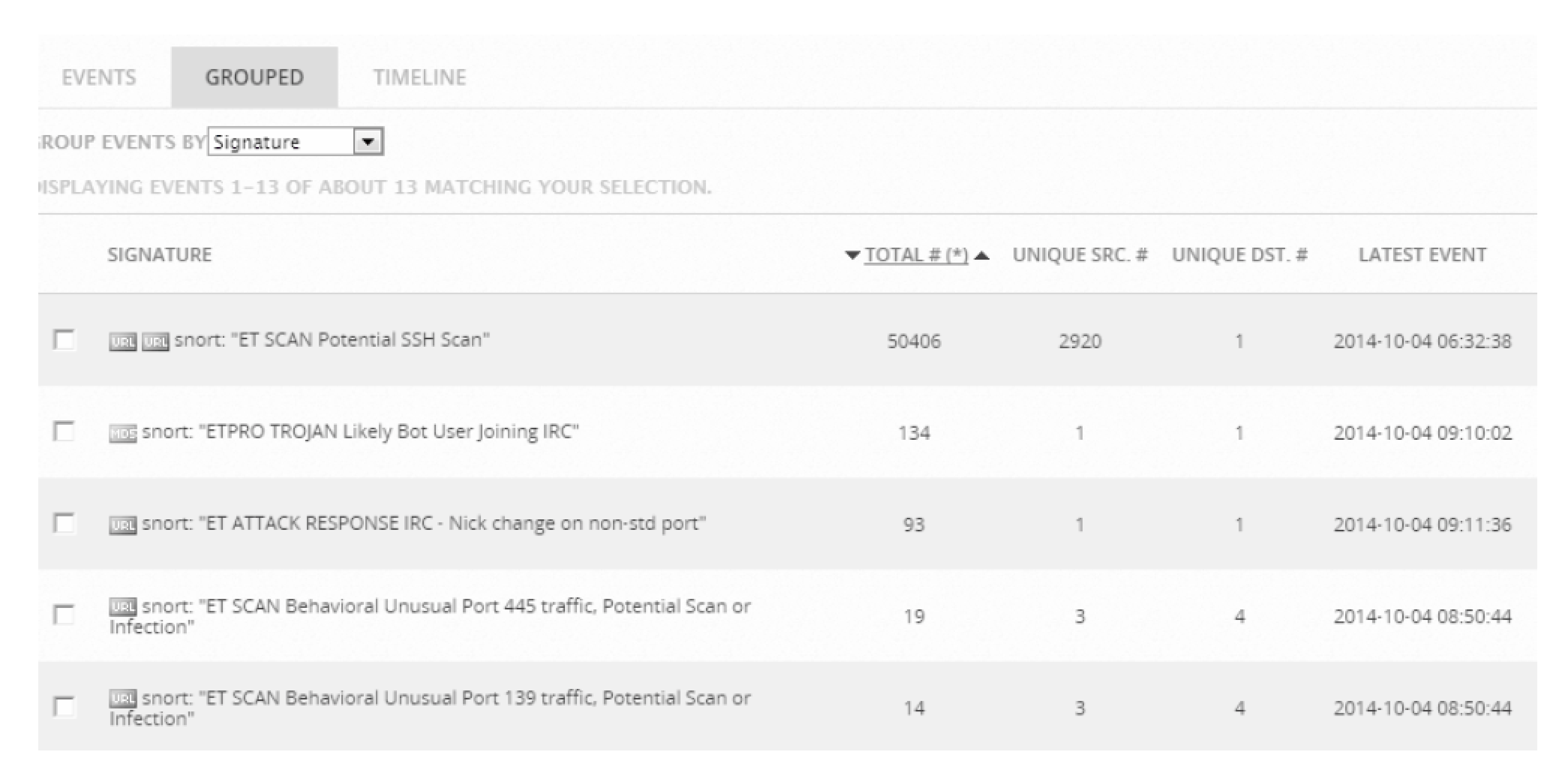


图 2-87　过滤出 Snort 事件报警

从图 2-88 中可以看出在 SIEM 控制台中，特征码为“snort:ET SCAN potential SSH Scan”的事件总共出现了 50406 次，OSSIM 系统可以归纳为一条报警，并记录到数据库。



SIGNATURE	TOTAL # (*)	UNIQUE SRC. #	UNIQUE DST. #	LATEST EVENT
snort: "ET SCAN Potential SSH Scan"	50406	2920	1	2014-10-04 06:32:38
snort: "ETPRO TROJAN Likely Bot User Joining IRC"	134	1	1	2014-10-04 09:10:02
snort: "ET ATTACK RESPONSE IRC - Nick change on non-std port"	93	1	1	2014-10-04 09:11:36
snort: "ET SCAN Behavioral Unusual Port 445 traffic, Potential Scan or Infection"	19	3	4	2014-10-04 08:50:44
snort: "ET SCAN Behavioral Unusual Port 139 traffic, Potential Scan or Infection"	14	3	4	2014-10-04 08:50:44

图 2-88　显示 Snort 过滤结果

对日志进行归一化处理后的消息头信息为如图 2-89 所示：

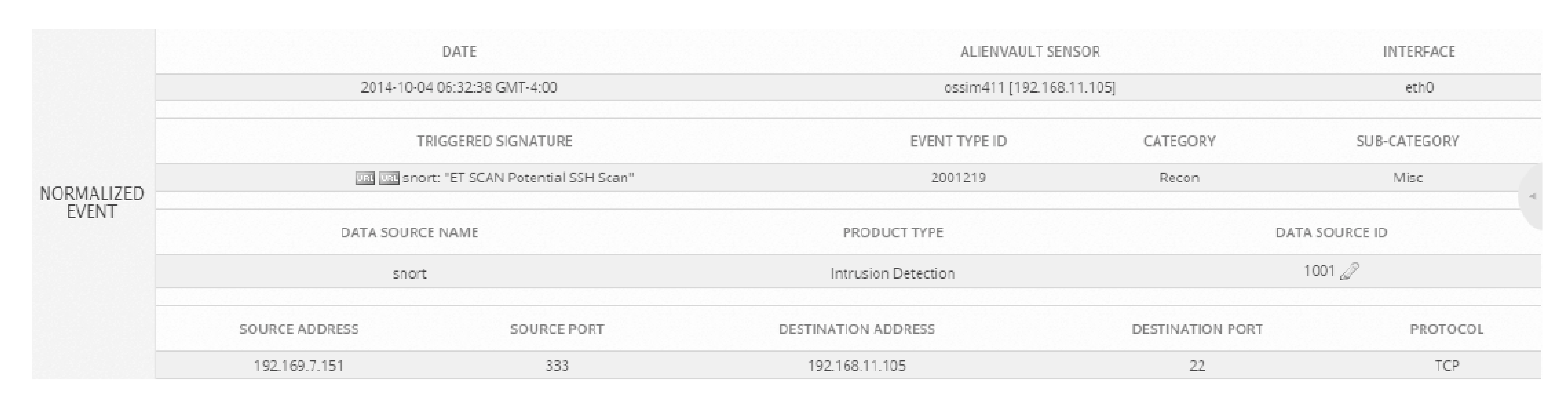


图 2-89　SIEM 归一化日志显示

上图中日志包含了时间戳（包括时区）、传感器、网卡、特征码、事件类型、数据源、源地址、目标地址、源端口、目标端口以及协议等信息。根据这些信息在成千上万条重复数据中找到规律，根据特征码的不同，将重复数据“消除”，这就是归纳合并（事件合并标准主要是参考事件类型、协议类型、目标 IP 地址）。

另外，我们还可以通过 SIEM 中提供的时间线（TimeLine）功能分析事件的发展趋势，通过列出 TOP N 的方式显示出趋势，所列举的趋势能够反映一段时间内的数据变化，例如 Siem vs Logger events 显示 24 小时内变化趋势、Ticket resolution time 显示了一周内变化趋势、Tickets closed by month 显示了每月统计趋势、Security events trend:last day 最近一天安全事件变化趋势、Security events trend:last week 最近一周变化趋势，更多变化趋势还可以到报表子系统中输出，如图 2-90 所示。将这些日志归纳出安全事件 Top5、嗅探主机 Top10，或者占用带宽最多的 Top10，常见事件类型 Top10 等，从这些信息中可以快速挖掘新的可疑信息。这对于分析网络攻击事件至关重要。

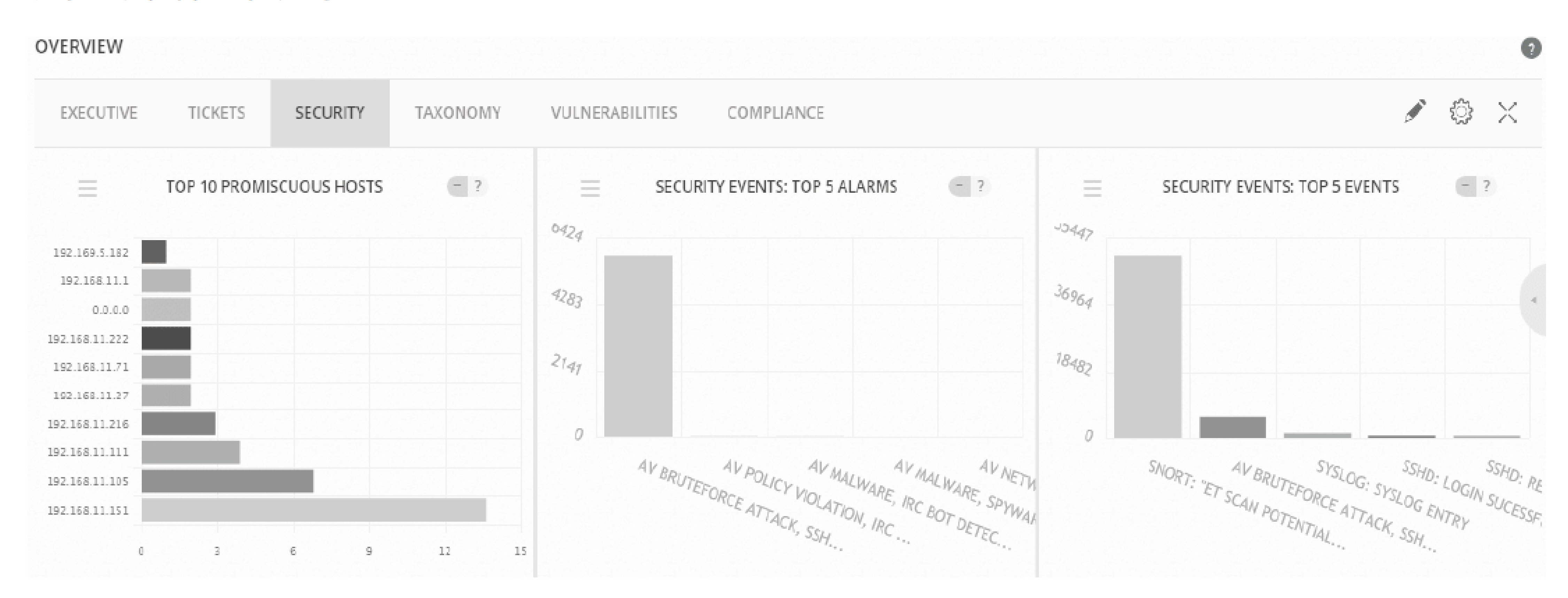


图 2-90　仪表板显示的网络报警和攻击排名的柱状图

2.16.2　将重要日志加入到知识库

OSSIM 系统中采用了共享信息的接口，称之为知识库（KDB），知识库是具有一定智能的信息安全管理软件，在开源领域目前只有 OSSIM 系统具有基于知识的系统。在此之前进行安全评估和漏洞分析时，经常会碰到对于海量信息查找缓慢，查找效率低下等问题，如今在 OSSIM 中使用知识库提高了我们分析问题的速度。知识库往往通过关联分析引擎联动，为应急响应提供了知识保障，流程图如图 2-91 所示。

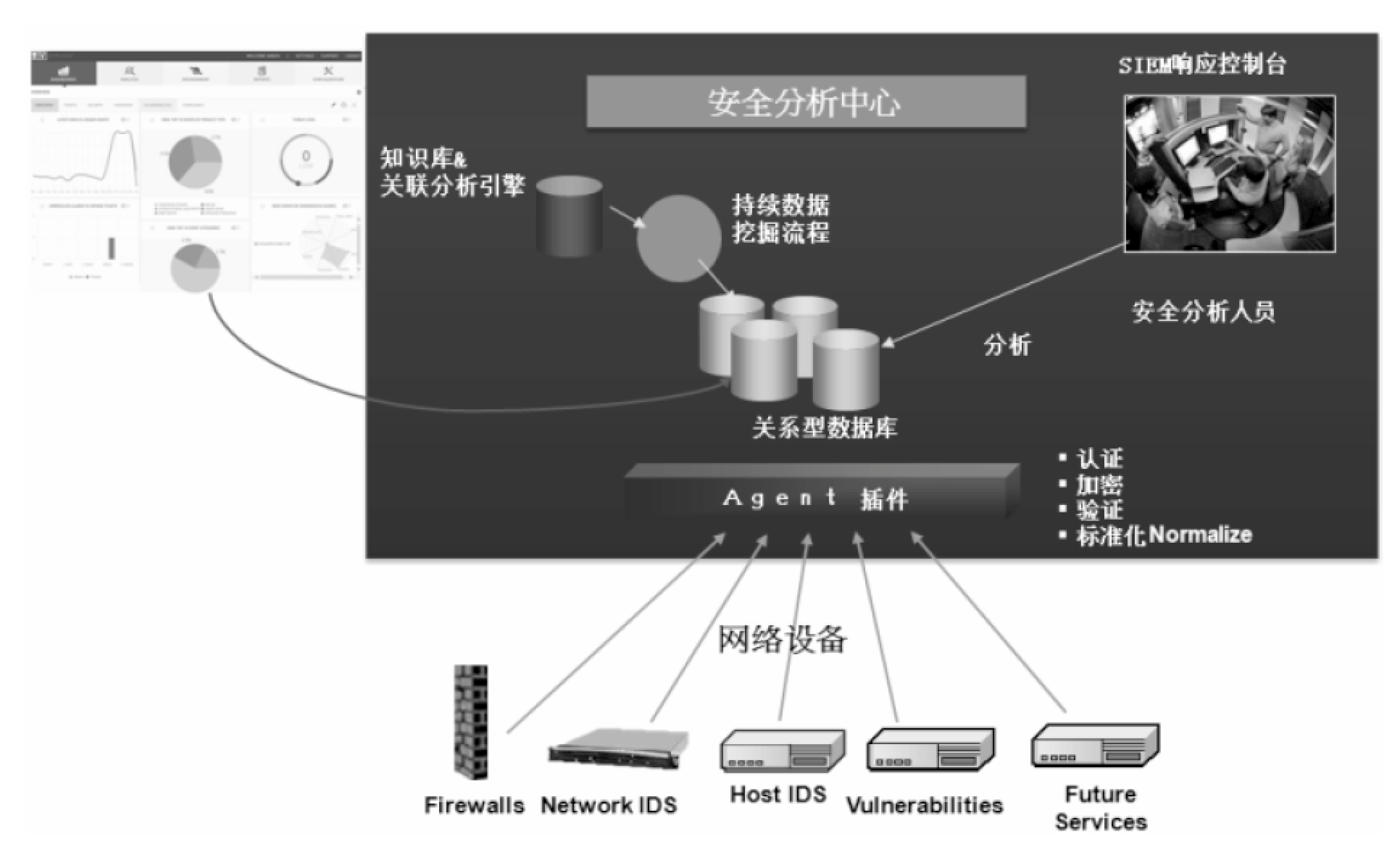


图 2-91　关联分析联动示意

应用之一是可以将漏洞扫描的结果作为 IDS 系统的数据源。知识库具体体现在哪些地方呢？我们打开 SIEM 控制台查看事件信息就能发现。如图 2-92 所示，下面这是 2 条 SSHD 认证成功的事件，我们单击任意一条可以发现。

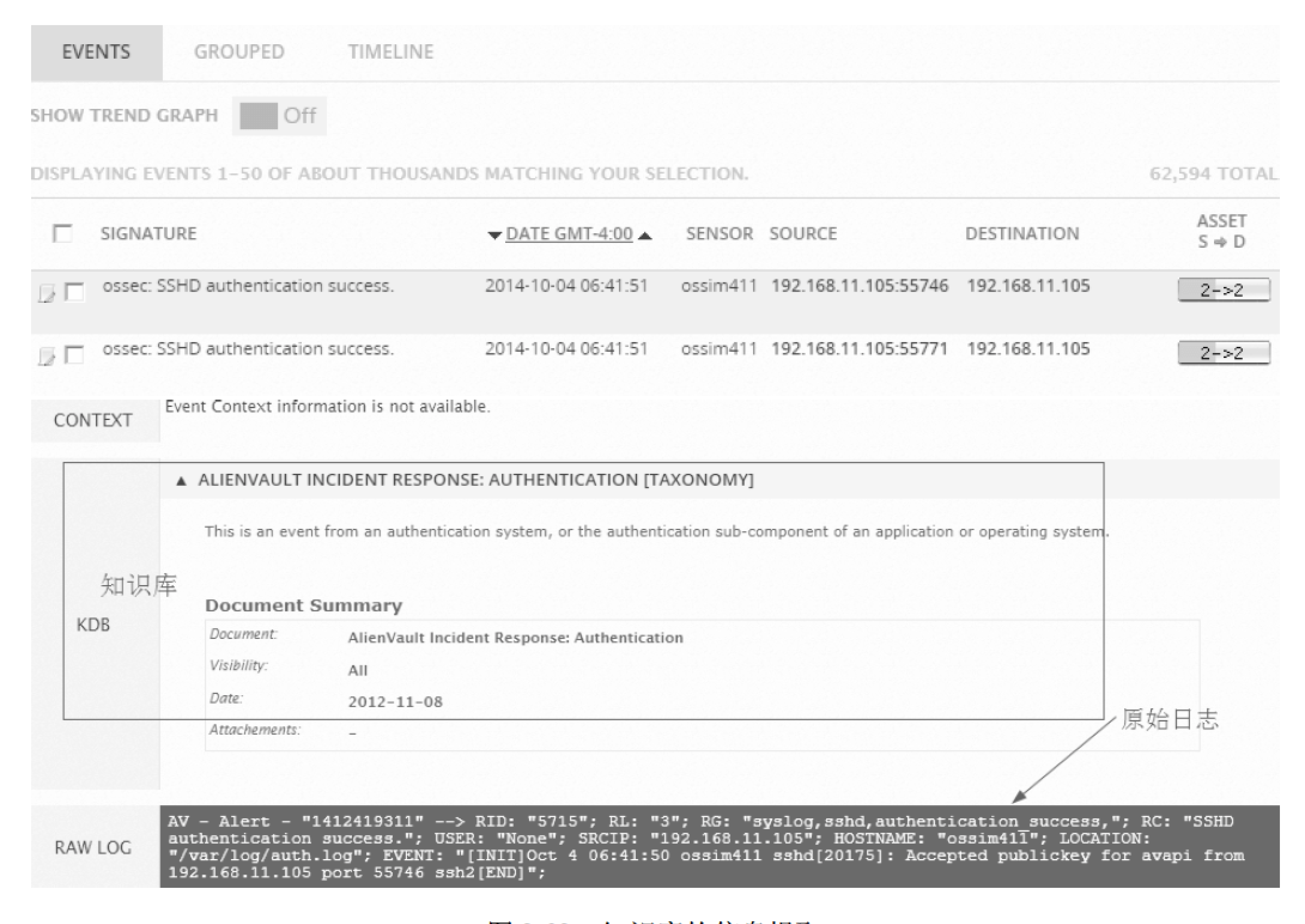


图 2-92　知识库的信息提取

图中的原始日志是日志规范化过程的一部分，原始日志是必须保留的。最重要的是可以从规范化日志中提取原始日志作为司法证据。下面我们详细看看它提供哪些具体信息，事件的下方是知识库信息，这些信息可由用户自己添加。

首先，通过选择日志左侧的方框选项，再单击右键就能加入到相应的知识库中，如图 2-93、图 2-94、图 2-95 所示。

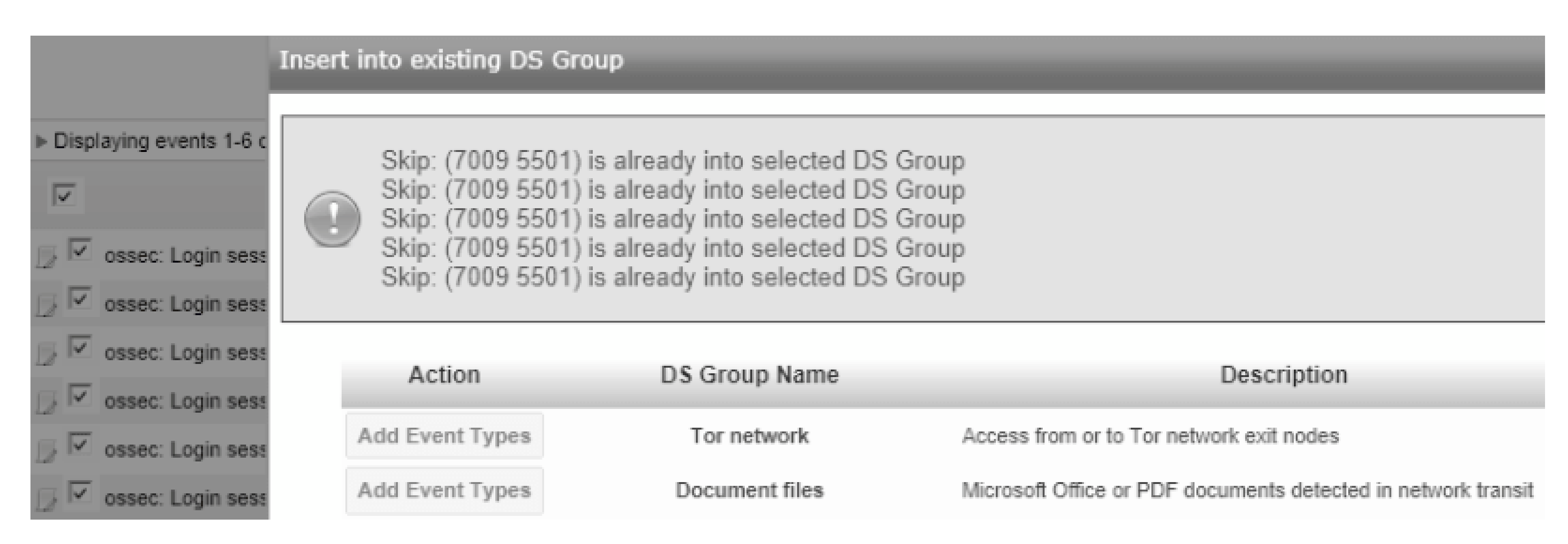


图 2-93　将日志加入数据源

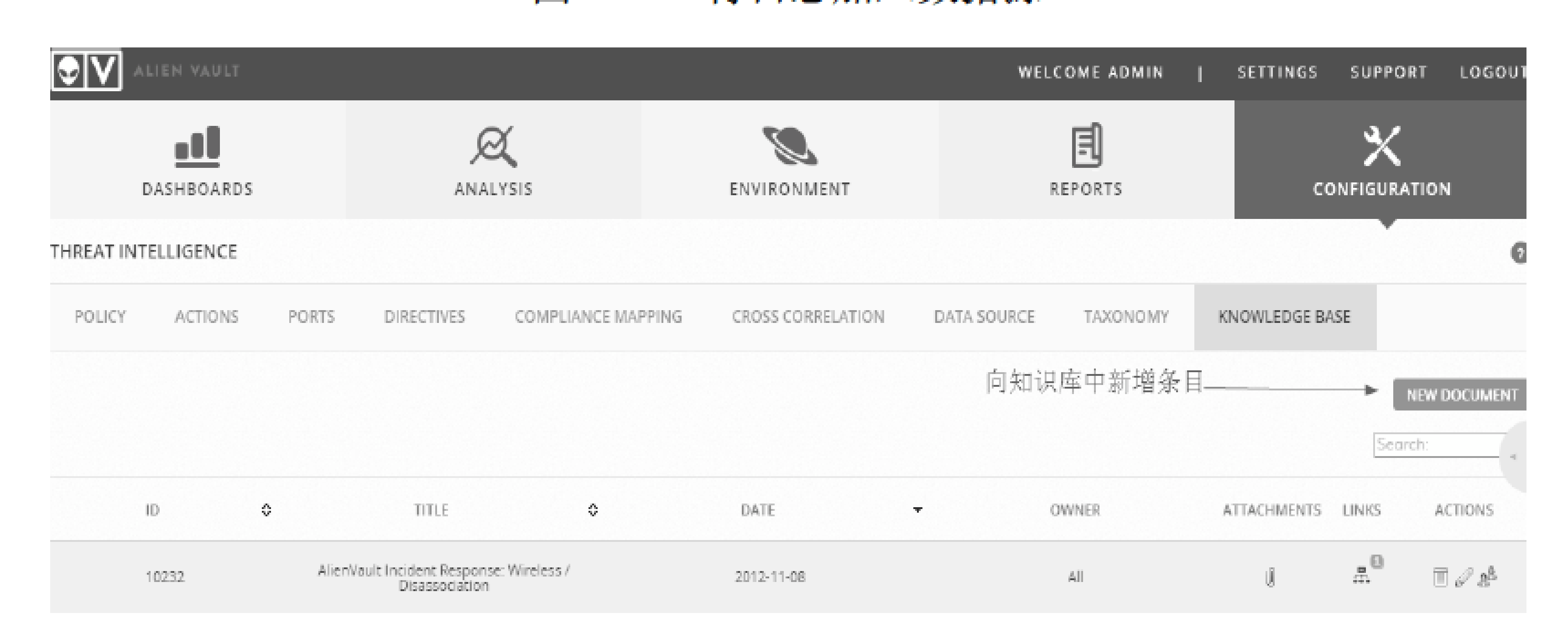


图 2-94　向 OSSIM 知识库中新增条目

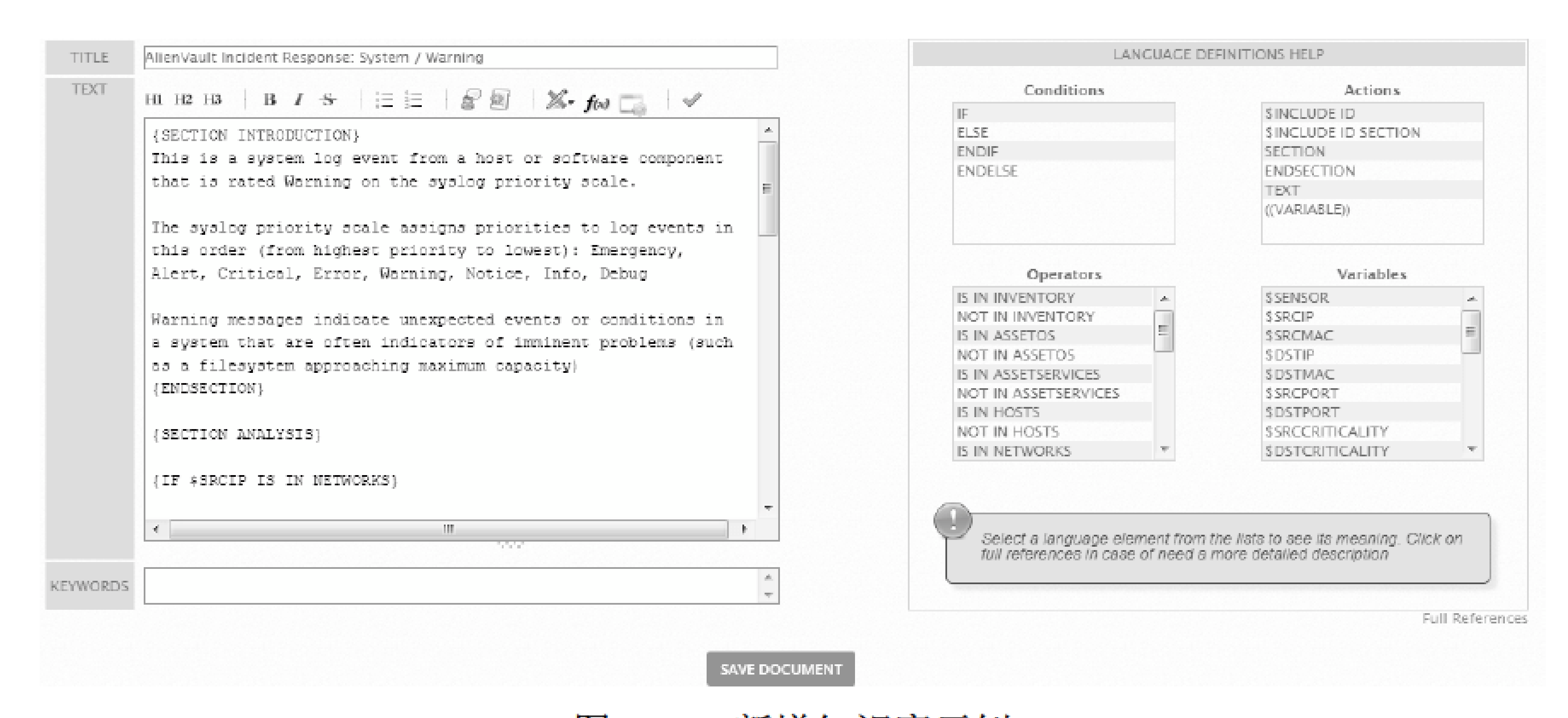


图 2-95　新增知识库示例

在新增到知识库中的条目中还能上传图片（系统故障截图），统一保存在/usr/share/ossim/uploads/目录下，文件名以新加文档的 ID 号命名。

通过收集这些日志信息，并存放到 OSSIM 的知识库中，经过长期积累记录就能够为企业积累一笔宝贵的财富，为今后网络故障诊断和取证留下重要的判定依据。

2.16.3 SIEM 中显示不同类别日志

在 OSSIM 系统 SIEM 事件收集中，常会收到重复日志，例如，攻击者对网络上的某主机进行端口扫描，在每次探测中，系统会创建单独的事件，每扫描一次 IDS 很可能创建单独的日志，而这些大量重复事件的源 IP 和源端口在每个事件中不同，而目标 IP 地址（被攻击的服务器）在每个事件中基本相同，但这些重复的数据对安全人员意义不大，管理员需要看到具有不同特征的事件序列。更重要的是某个事件重复了多少次，频率是多少。在 OSSIM 的 Web UI 中用图形化方式直观显示的攻击目标端口（TCP、UDP），路径为 Dashboards→Overview→Security，如图 2-96 所示。

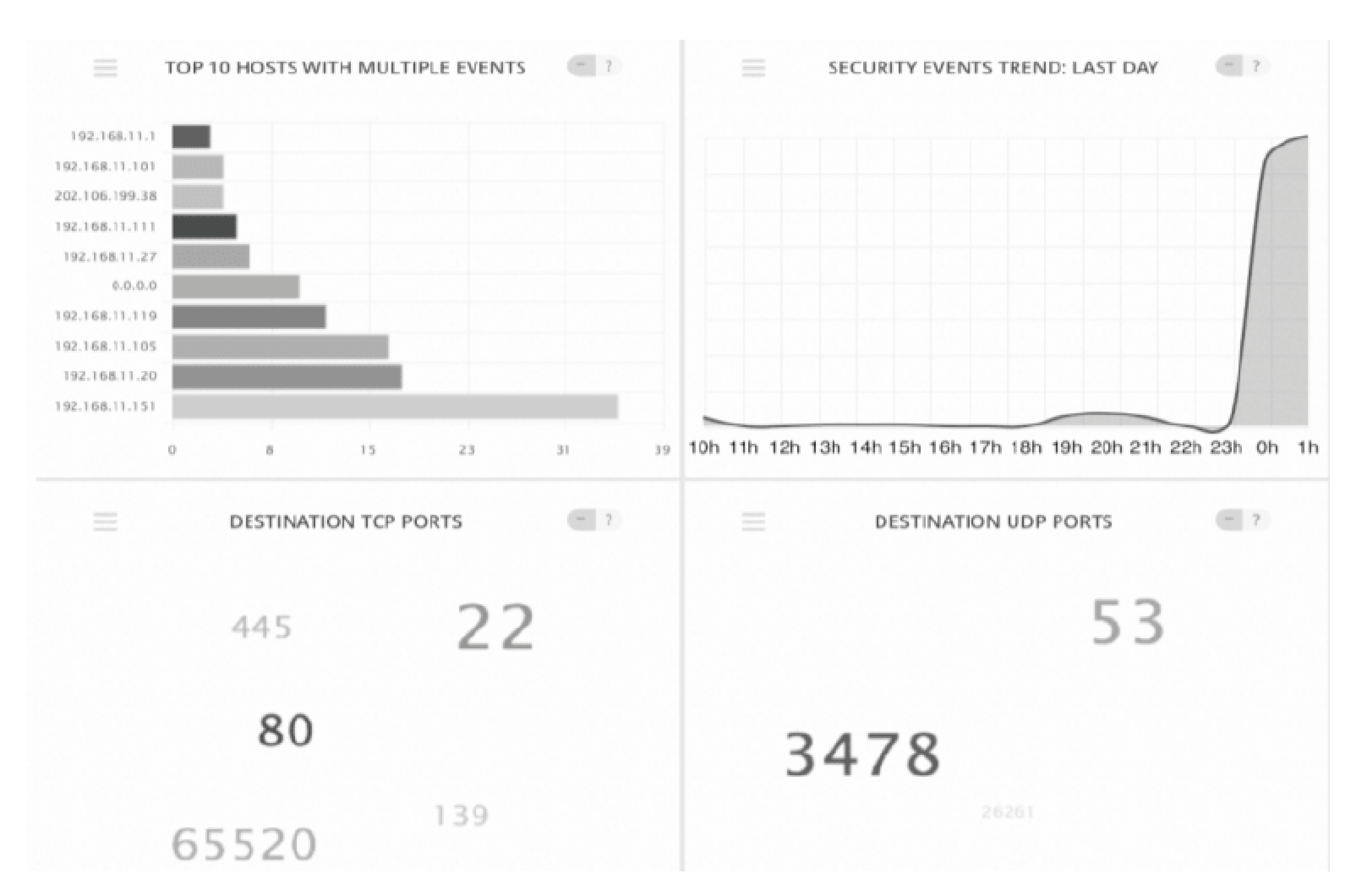


图 2-96 图形化显示目标端口

OSSIM 系统中会根据日志的特征码进行自动分类，还可以统计源地址和目标地址的数量，值得注意的是，源 IP 地址不足以确保唯一性，很可能是伪造的，而目标地址通常在攻击期间保持不变，这有助于安全人员结合特征码来分析故障。如图 2-97 所示，这个截图列举了 OSSIM 4.6 系统中的 SIEM 消除重复事件的显示结果。默认按特征码分组，如图 2-98 所示。

图 2-97 通过 Grouped 显示不同类型事件

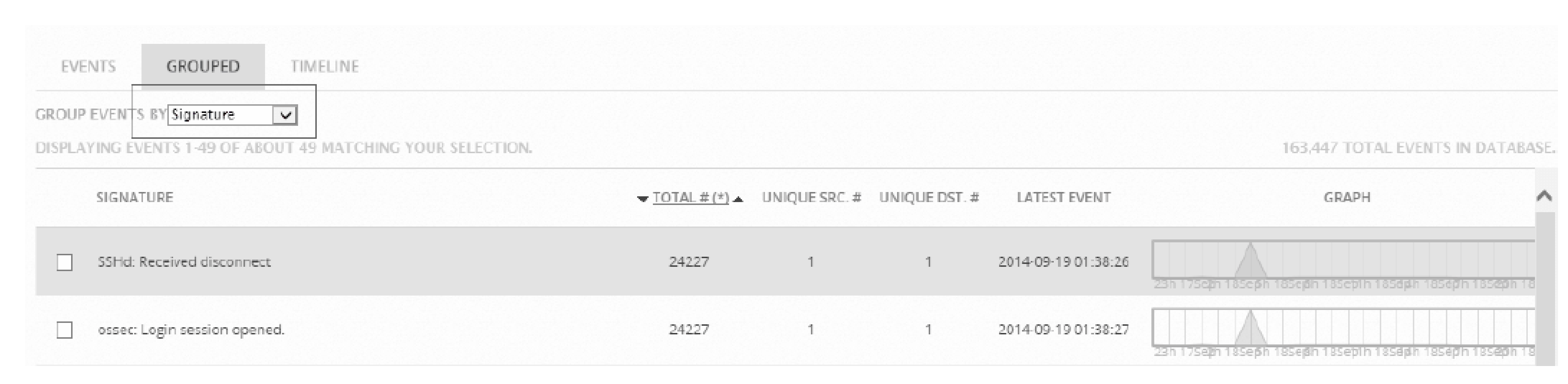


图 2-98　根据特征码来分组

在 SIEM 控制台中具有强大的筛选功能，可以按不同事件（Unique Events）、不同的地址（源地址和目标地址）以及不同的国家显示，如图 2-99 所示。注意在选择 Grouped 时，系统会对 alienvault_siem 库中 acid_event 表进行 SELECT 操作，当某类事件数量达到千万级时，等待时间较长。

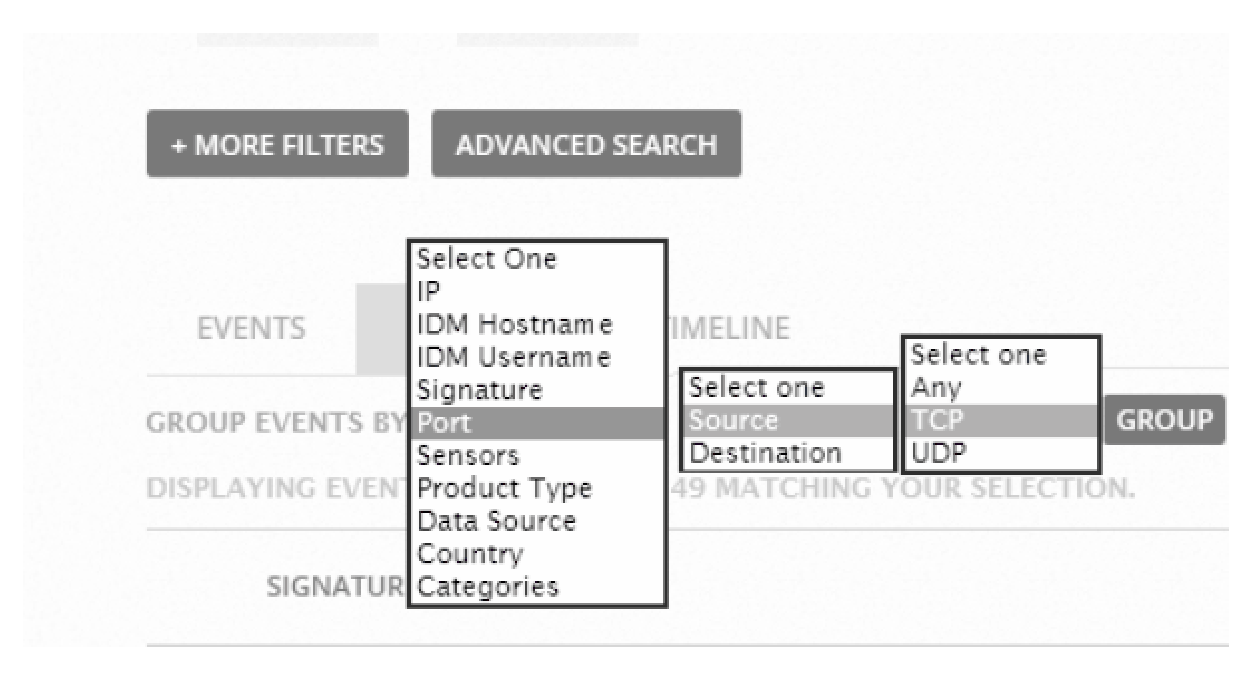


图 2-99　SIEM 控制台筛选事件

下面我们举一个应用例子：根据源 IP 进行定位显示。如图 2-100 所示。

图 2-100　根据源 IP 定位显示

我们选取第一个搜索的源 IP：10.232.63.76，会显示如图 2-101 所示的相似内容。

图 2-101　显示某 IP 详细属性

这里显示了非常详细的信息，这些信息当中主要是针对检测公网地址才起作用。在这张图中，我们以顺藤摸瓜的方式在“External”栏中可找到很多有关 IP 的背景资料，这些资料是来自于不同组织对这个 IP 检测的测评结果，对我们判断一个陌生 IP 的信誉度，有着非常重要的作用。

2.16.4　常见搜索信息

下面介绍一些网络安全人员应该关注的搜索类型：

- SSH 服务 N 次登录之后，仅一次登录成功，这很有可能是尝试某种暴力破解，此时我们可以通过 OSSIM 策略进行报警，这里 N 的范围一般为 4 ~ 6。
- VPN 用户如果在非工作时间登录（尤其是夜晚或凌晨），这表示可能受到攻击，管理员必须警觉。
- 网段内某台主机开始探测其他主机，这台主机有可能受到蠕虫或感染恶意软件，也有可能受到网络攻击，这时需要在 OSSIM 的 SIEM 中搜索防火墙日志，进一步分析。
- 在极短时间内，N 次尝试访问共享文件夹（或者文件），这很可能用户账户泄露，攻击者正在入侵，往往这个时间是非工作时间。
- 交换机配置在非工作时间被修改，意味着很有可能被攻击。
- Web 服务器短时间出现很多 Web 404、401、500 错误代码，表明服务器工作异常。

2.16.5　仪表盘显示

在 OSSIM 4 系统中，主界面增添了自定义面板功能选项，单击铅笔状按钮，用户可以自己添加 Honeypot Activity、Network 等功能。如图 2-102 所示。

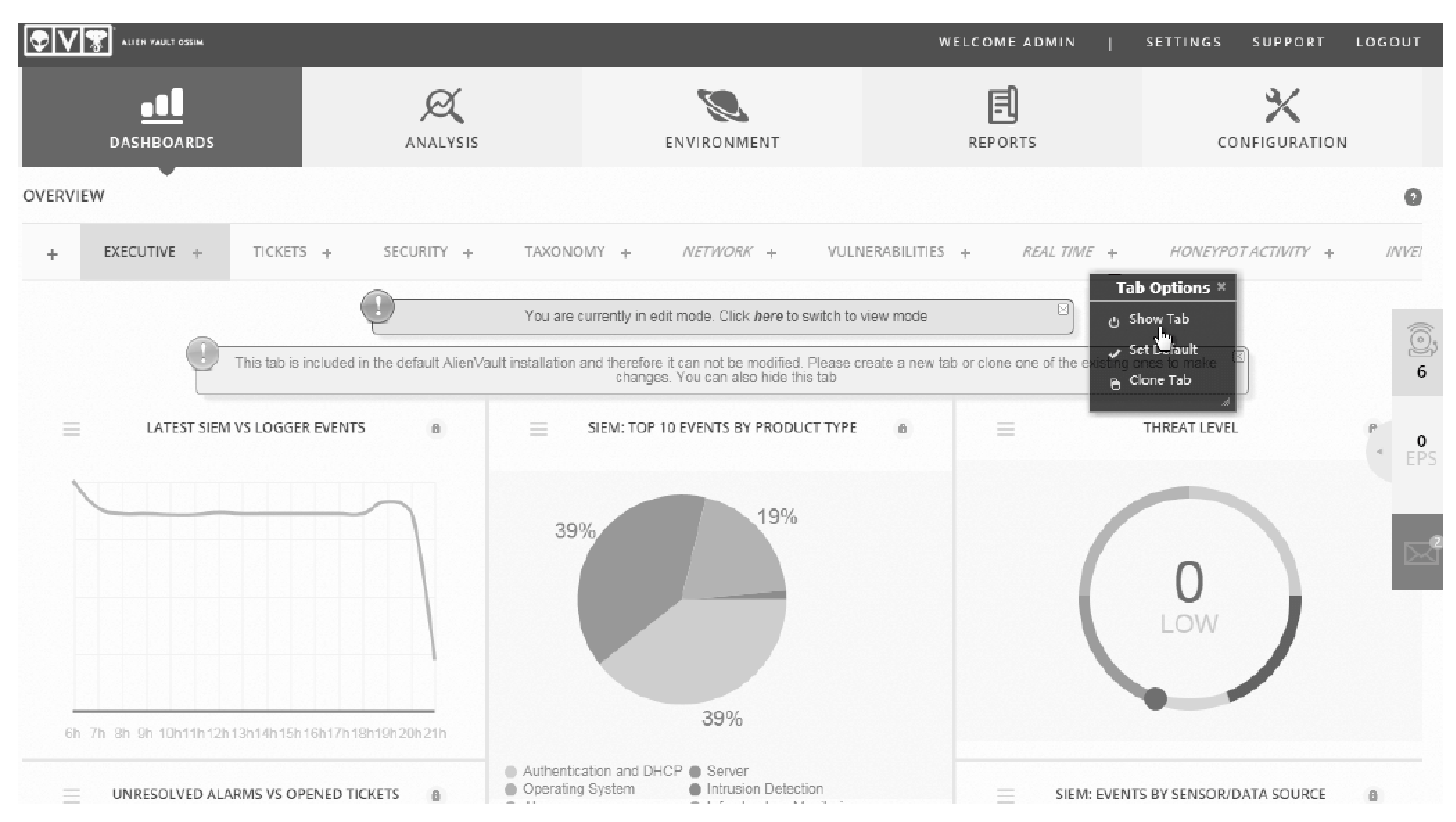


图 2-102　调整仪表盘显示内容

另外，用户还可以单击 SIEM 控制台中右侧的书本状图标后，自定义 SIEM 显示内容，如图 2-103 所示。

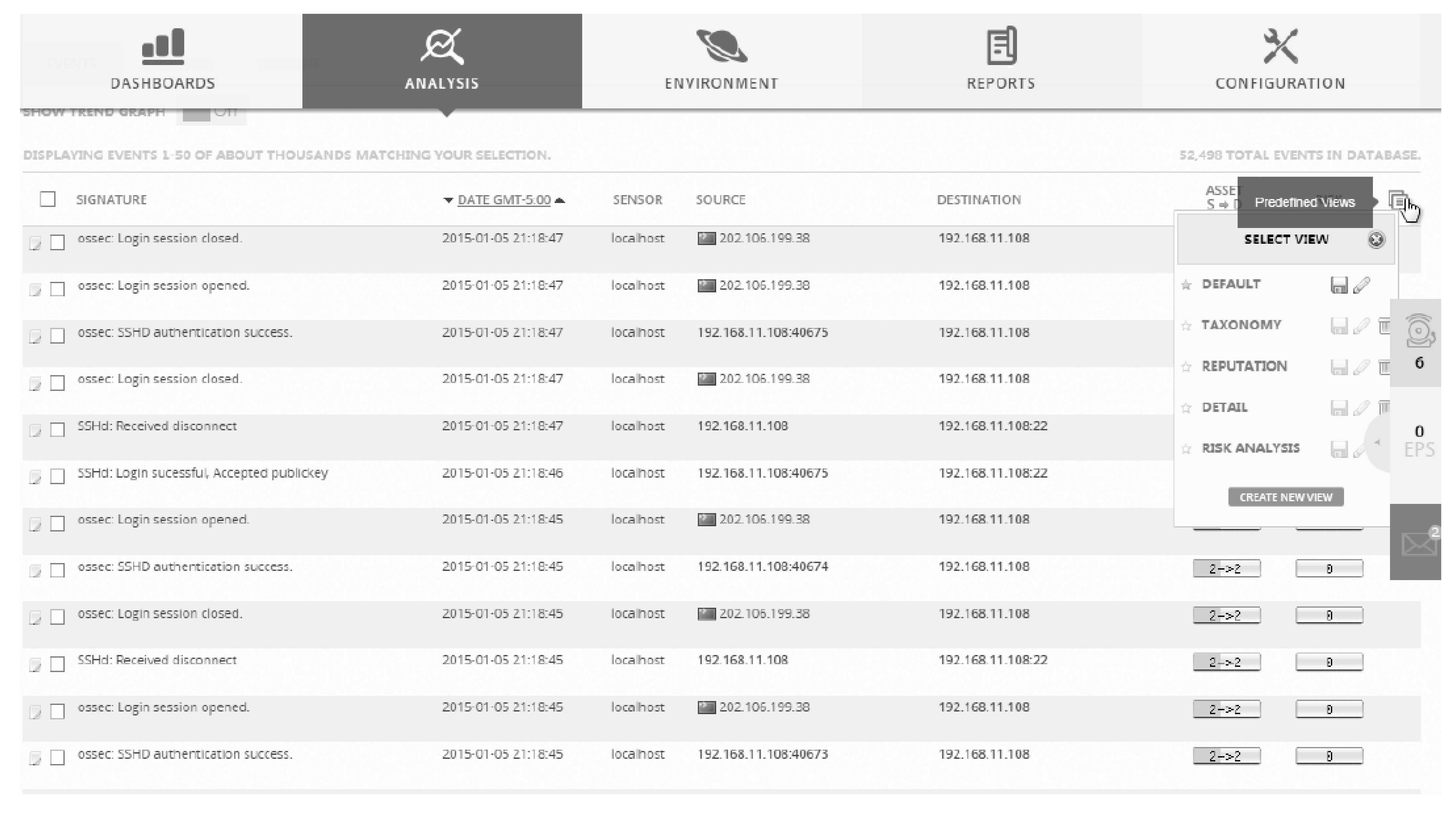


图 2-103　自定义 SIEM

自定义之后，增加了 Payload（有效负载），显示效果如图 2-104 所示。

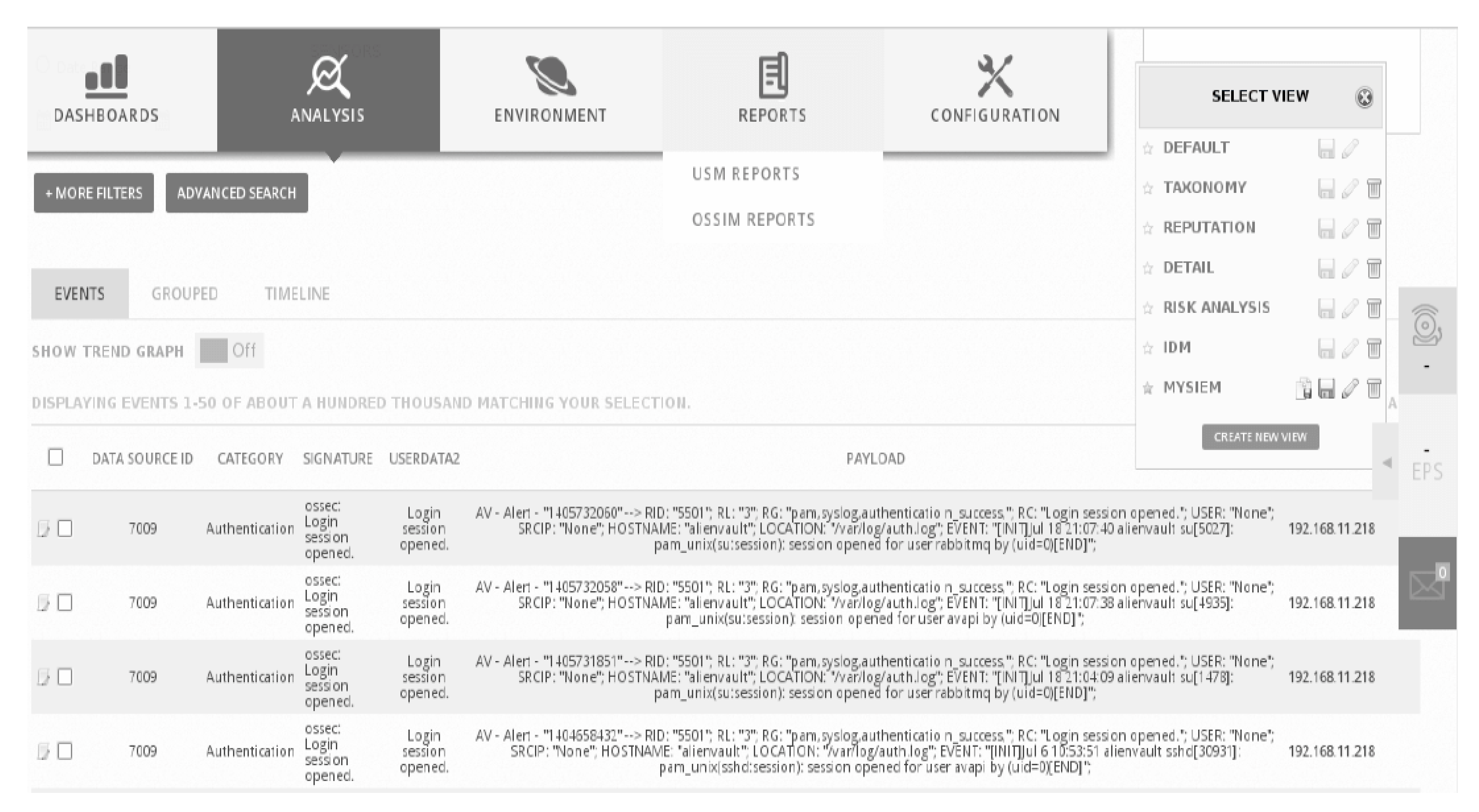


图 2-104　自定义 SIEM 效果

2.16.6　事件删除与恢复

为了在 SIEM 控制台中删除不必要的干扰事件，读者需要掌握 3 个重要按钮的功能。

- Delete selected：删除勾选项事件，为了快速勾选所有事件，可将 Signature 前面的复选框选中。
- Delete all on screen：该选项可快速删除当前屏幕显示的事件。
- Delete entire query：这个很危险，能够删除全部查询。

在系统负荷较大或事件数量大时，删除过程会变得缓慢，数据执行大量 INSERT.DELETE 使 CPU 占用率很高，在 Search criteria（搜索条件）方框下方会出现“Deleting in backgroud（后台删除中）”字样，大家不要重复单击删除按钮。

OSSIM 系统每天会自动备份 SIEM 事件，存储路径为/var/lib/ossim/backup/，文件名为 environment_database_backup_日期_tar.gz，其中“日期”采用年、月、日表示。当找到备份文件后，不必在命令行手动导入，只要在 Web UI 中轻点鼠标就能完成。下面我们尝试恢复 SIEM，首先在 Configuration→Administration→Backup 菜单中选择 Dates to restore 中的某个日期，比如 05-04-2015，然后单击 RESTORE 按钮，随后系统开始恢复。如图 2-105 所示。

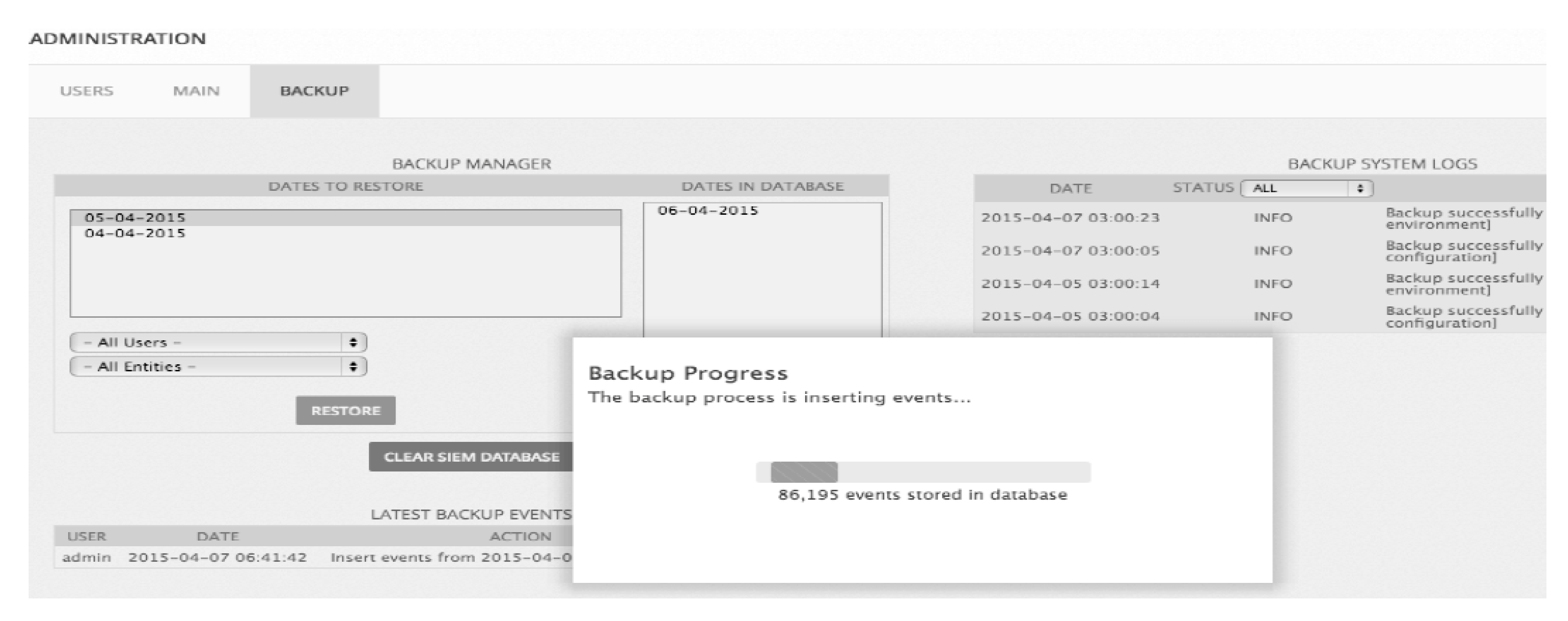


图 2-105　备份进度

如果使用 OSSIM 4.3，那么在这个备份目录下的文件为 ossim-backup_年+月+日.sql.gz

2.16.7　深入使用 SIEM 控制台

用于分析和处理 Snort 收集的入侵数据并以图形化方式展示出来的主要工具就是 SIEM 控制台，SIEM 控制台以一种比 Snort 输出的原始数据更加容易理解的方式给出报警和入侵数据，如图 2-106 所示。

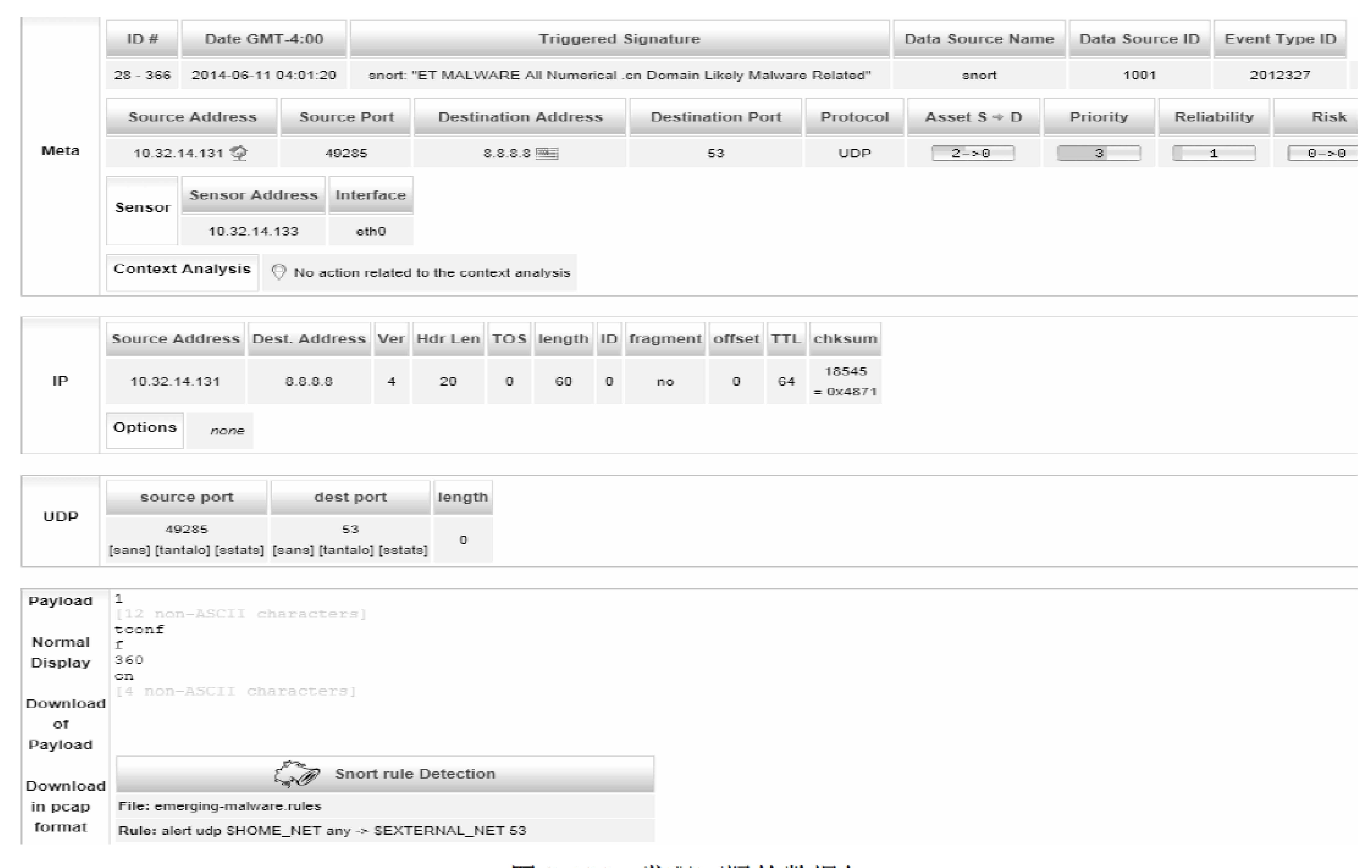


图 2-106　发现可疑的数据包

数据按事先设计好的逻辑方式排列，这样有助于安全员快速决策。数据包以易于理解的方式展示出来，这样可以很清楚地记录了包中承载的信息。

除此之外，还能够清晰展示数据包头信息，如图 2-107 所示。这相当于同时运行了 Wireshark 抓包工具。

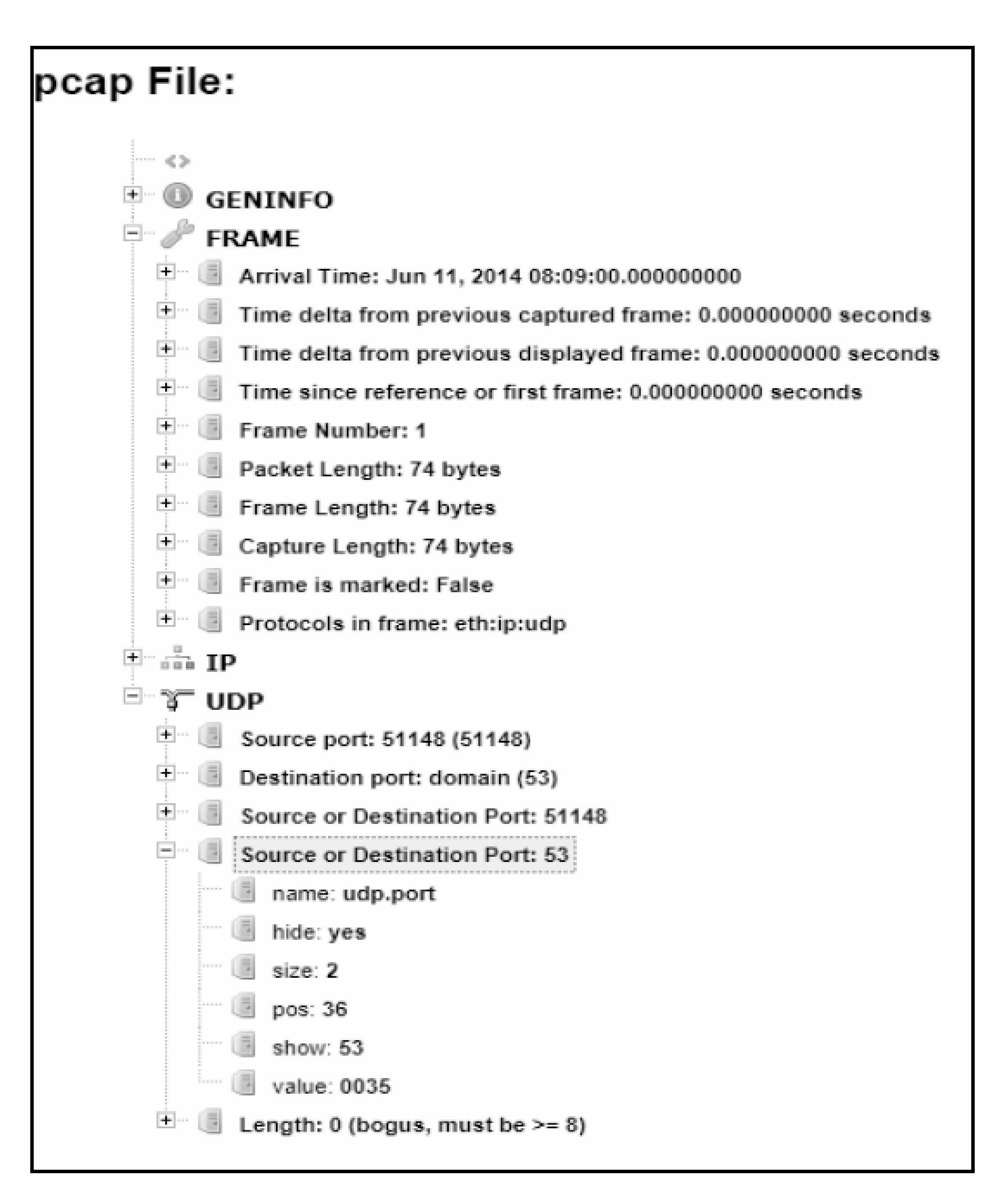


图 2-107 包头信息

由于性能的原因，不可能将含有过多特征的文件信息存储在数据库里，在 OSSIM 控制台中每次显示报警时，把特征相对应的外部链接也显示出来。如果希望详细了解某个报警，则可以到专门的 URL 链接上查询。我们看个例子，如图 2-108 所示：

☐	snort: "ET SCAN Behavioral Unusual Port 445 traffic, Potential Scan or Infection"	1020	1	140	2014-06-11 11:15:01
☐	Apache: Moved Temporarily	647	2	1	2014-06-11 11:14:13
☐	snort: "ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port Reserved Bit Set - Likely Kazy"	321	1	2	2014-06-11 09:29:50
☐	ossec: System user successfully logged to the system.	57	1	1	2014-06-11 11:01:06
☐	snort: "ET P2P eMule Kademlia Hello Request"	53	1	1	2014-06-11 06:39:30
☐	Apache: Not Modified	35	1	1	2014-06-11 07:32:01
☐	snort: "ET SCAN Behavioral Unusual Port 135 traffic, Potential Scan or Infection"	23	1	8	2014-06-11 11:07:33

图 2-108 Snort 报警显示 URL

在第一行显示的 Snort“ET SCAN Behavioral Unusual Port 445 traffic，Potential Scan”出现

频率很高，达到 1020 次，在它前方有四个 URL 连接。

第一个图标链接到：http://doc.emergingthreats.net/2001569。显示内容如图 2-109 所示。

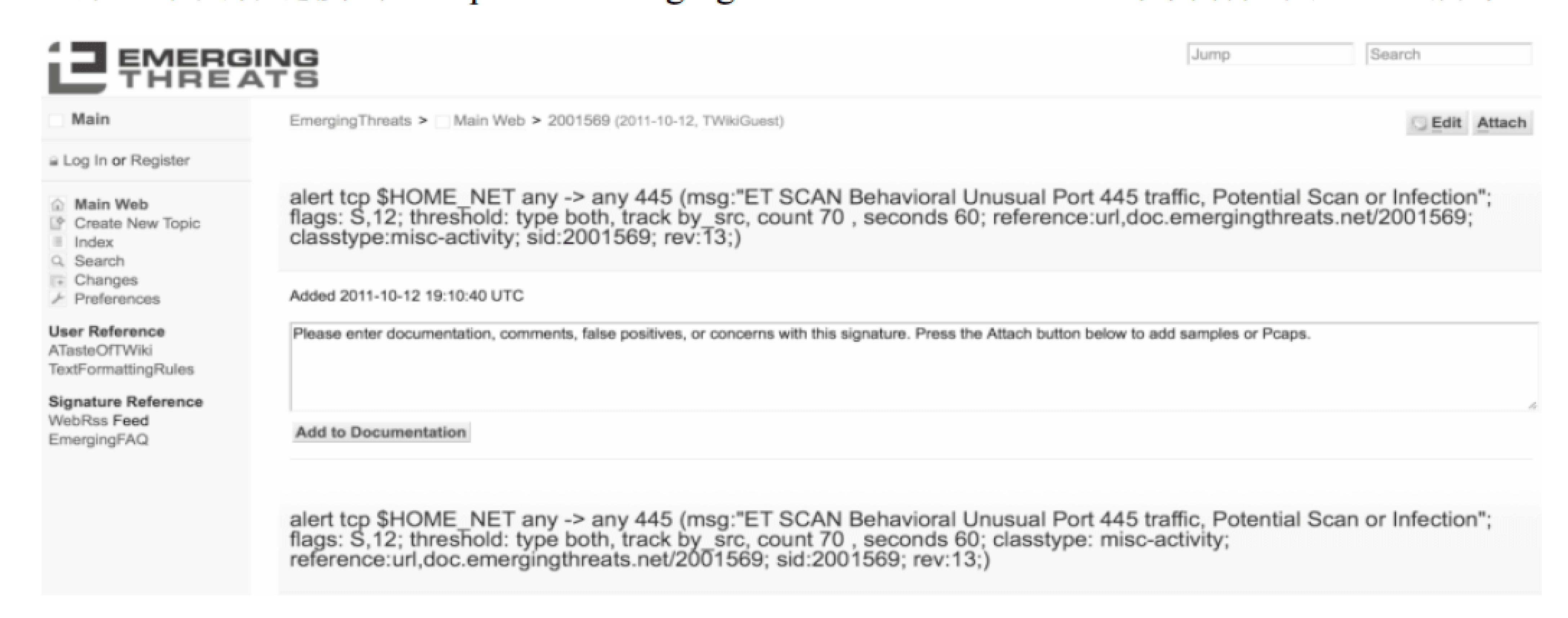


图 2-109　URL 显示内容

第二个图标链接到：http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-7172。

第三个图标链接到：http://doc.emergingthreats.net/2004242。

第四个链接到：http://doc.emergingthreats.net/2008907。

接下来，再看个例子，如图 2-110 所示：

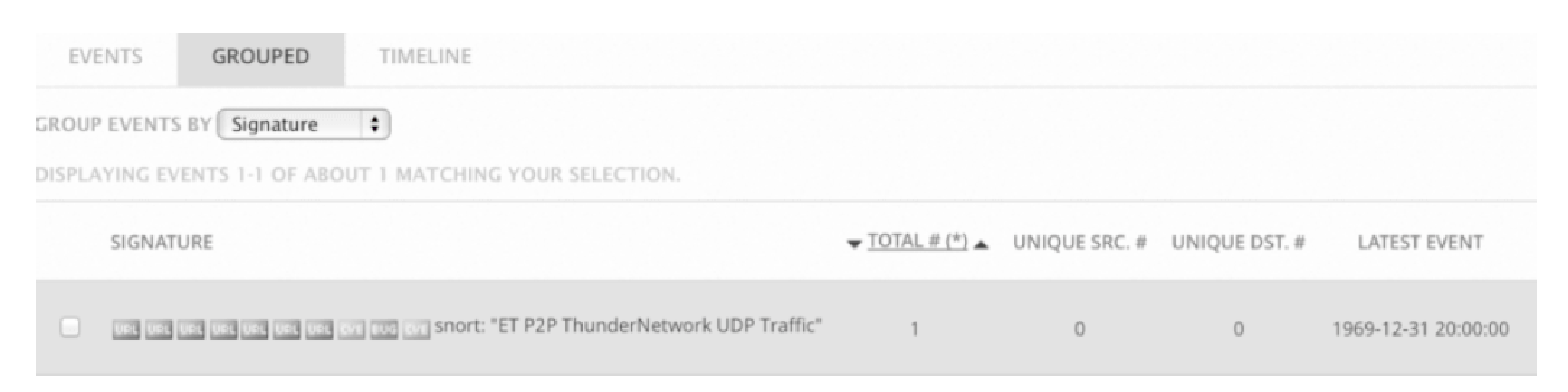


图 2-110　URL 显示

第一个图标链接到：http://doc.emergingthreats.net/2009099。

第二个图标链接到：http://en.wikipedia.org/wiki/Xunlei。

第三个图标链接到：http://www.kankan.com。

第四个图标链接到：http://www.exploit-db.com/exploits/12369/。

第五个图标链接到：http://securityhome.eu/exploits/exploit.php?eid=17879866924d479451d88fa8.02873909。

第六个图标链接到：http://secunia.com/advisories/43137/。

第七个图标链接到：http://www.exploit-db.com/exploits/16087/。

第八个图标链接到：http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-4620。

第九个图标链接到：http://www.securityfocus.com/bid/44638。

第十个图标链接到：http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4091。

这些地址是如何定义？我们可以在 Web UI 的 Configuration→Threat Intelligence→Date Source 下方的“Manage Reference”按钮中找到答案。

我们可以在系统中修改这些默认的 URL，通过单击“Manage References”按钮实现，在 OSSIM 的 Web UI 下通过 Configuration→Threat Intelligence→Data Source 菜单中单击“Manage References”按钮可以显示。如图 2-111 所示。

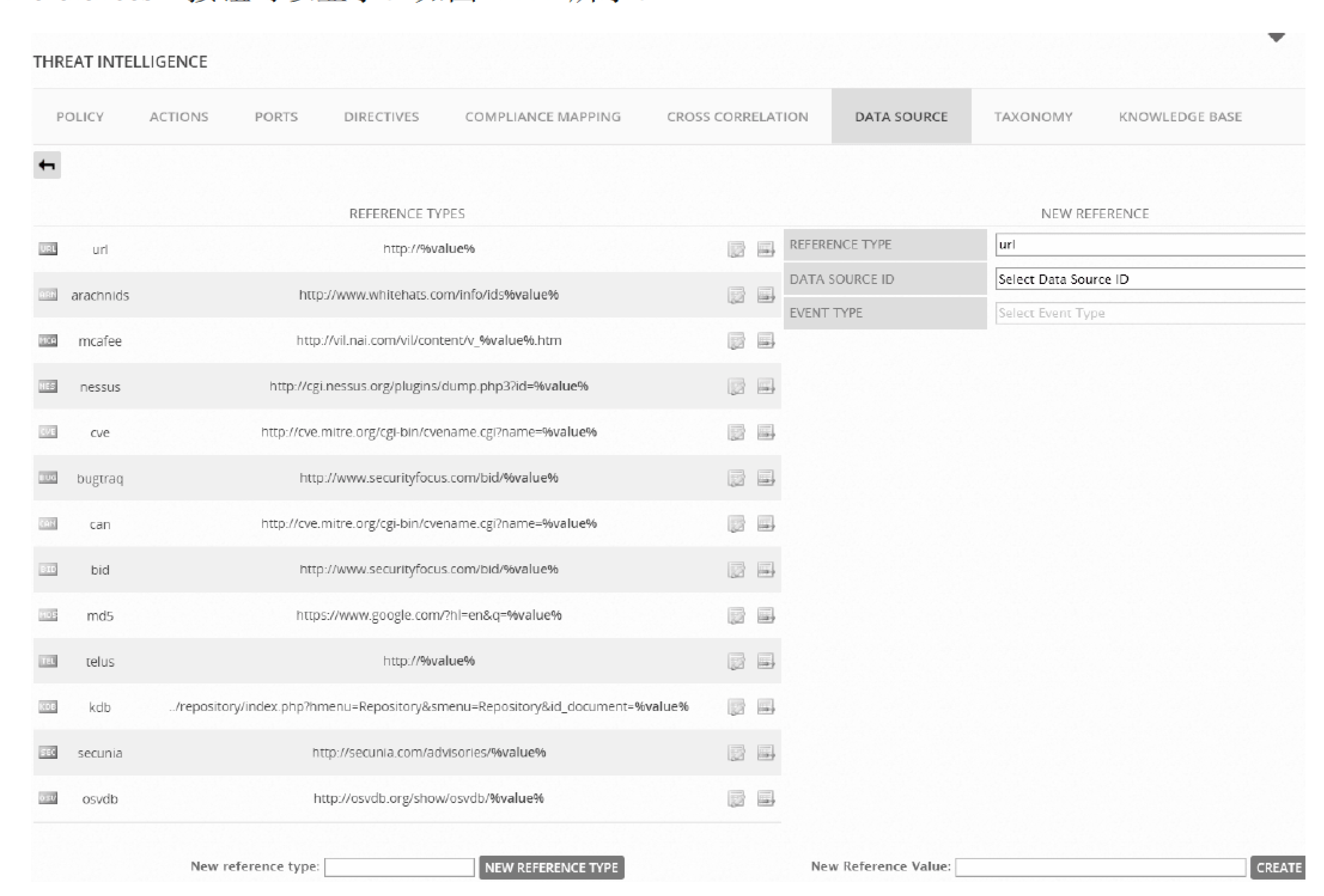


图 2-111　默认 References 定义

2.16.8　SIEM 事件聚合

SIEM 控制台可根据不同数据源，将报警事件进行有效聚合。冗余报警事件的聚合系统主要有前端数据采集、中间预处理和后端聚类与合并三大部分。其中前端数据采集模块主要有主机监控、网络监控、防火墙。中间的预处理模块主要是 Sensor 的插件对报警事件解码、分类和格式处理等。后端的聚类与合并分析就是在上面的基础上进行操作。

完成所做的分析处理之后，在 OSSIM 中使用主机监控软件 OSSEC 来实现对主机的审计记录、系统日志和应用程序的采集分析。它支持文件的完整性监控、注册表监控、端口的监控、Rootkit 检测和部分进程监控，同时还使用 Snort/Suricata 对网络上的数据量进行嗅探，可以完成对网络上的 IP 包进行测试等功能，也可以进行协议分析、内容查找和匹配，以用来探测多种攻击和嗅探（如缓冲区溢出和 CGI 攻击等）。

SIEM 事件聚合流程如图 2-112 所示。

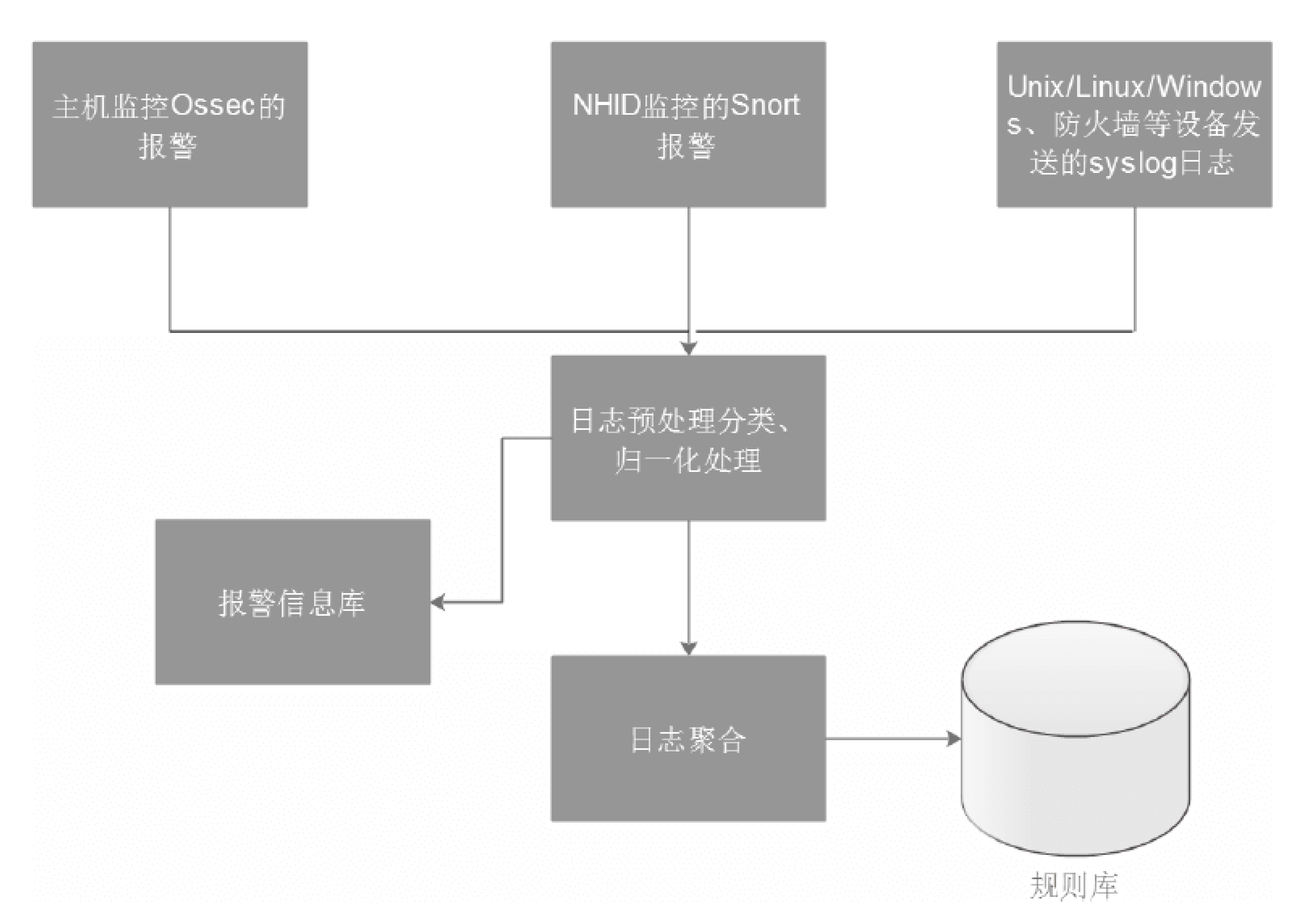


图 2-112　SIEM 事件聚合流程

2.16.9　SIEM 要素

前面介绍过不同版本 OSSIM 的 SIEM 控制台，它们虽然在 Web UI 有所差异，但是所具备的功能却相同，读者主要了解元数据（Meta Data）、IP 头数据、网络层协议和有效载荷（Payload）这几个概念。下面详细说明这些概念要素。

（1）元数据（meta data）

元数据是一种定义性数据，这些数据提供了有关 Snort 收集的入侵检测的数据信息。例如，记录流量过程的时间信息，对于整个攻击而言很重要，这也是事件关联分析的重要参数。但这种时间变化的趋势中出现的信息，并不在捕获包中出现。利用元数据标准可搜索到 Snort 收集所有包含标准的数据。

（2）传感器（sensor）

对于分布式的 OSSIM 系统而言有两个以上的 Sensor，这些传感器被部署在不同的网段，我们可以方便地查询到不同传感器所收集的数据。此外还有一个要注意的问题，同样一个传感器，放置在网络外部与放置在网络内部，对于同一个报警，所代表的含义是不同的。例如，放置在防火墙外侧，它如果收到了 Windows 共享访问试探，这就是疑似网络攻击。如果在内部网段部署的探测器，同样出现了 Windows 共享访问，那很有可能是一次正常的文件共享。

如图 2-113 所示，从使用中发现，Data sources 数据源的内容和插件有关，而插件在 Alienvault→Center 的 Sensor 中可以添加。

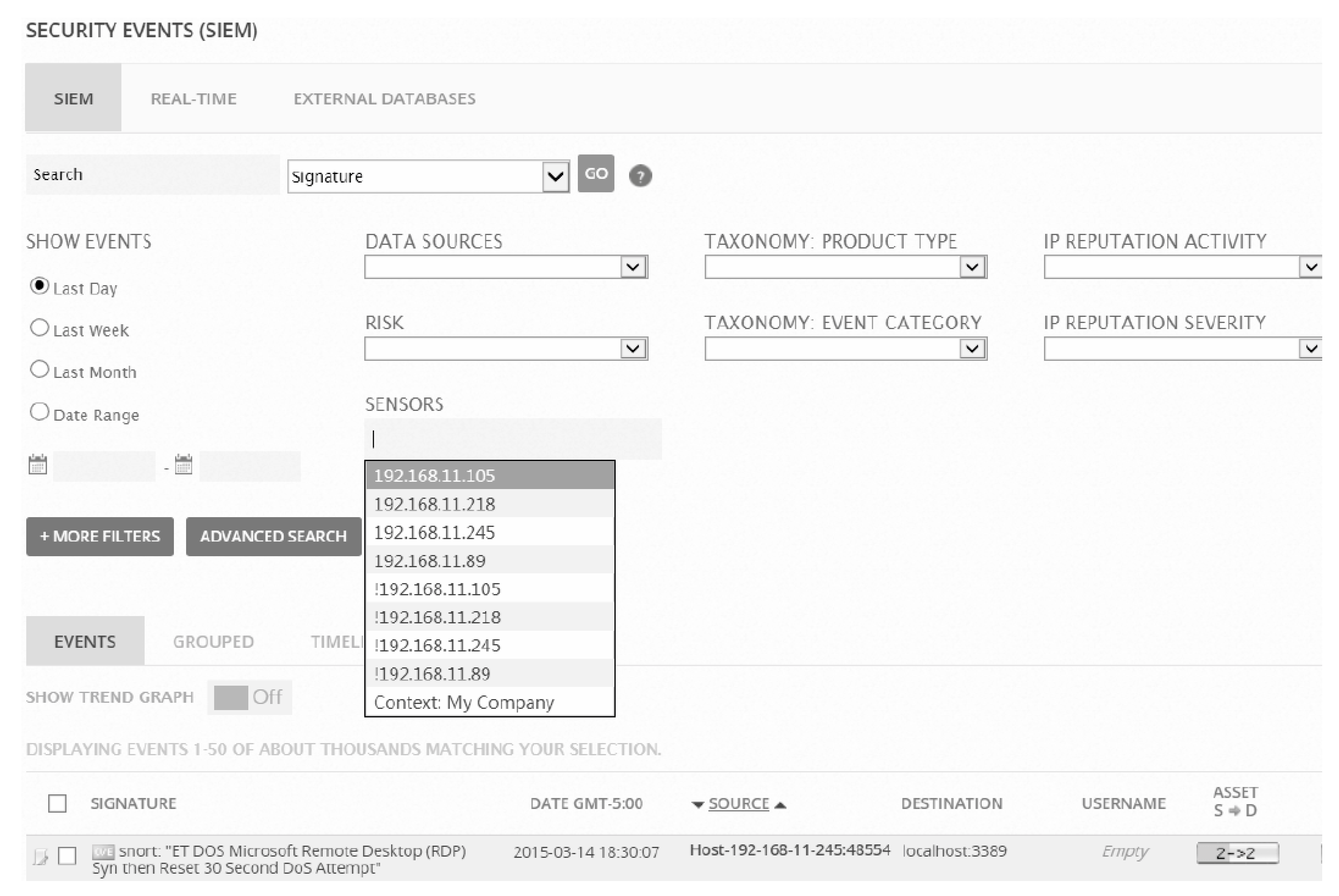
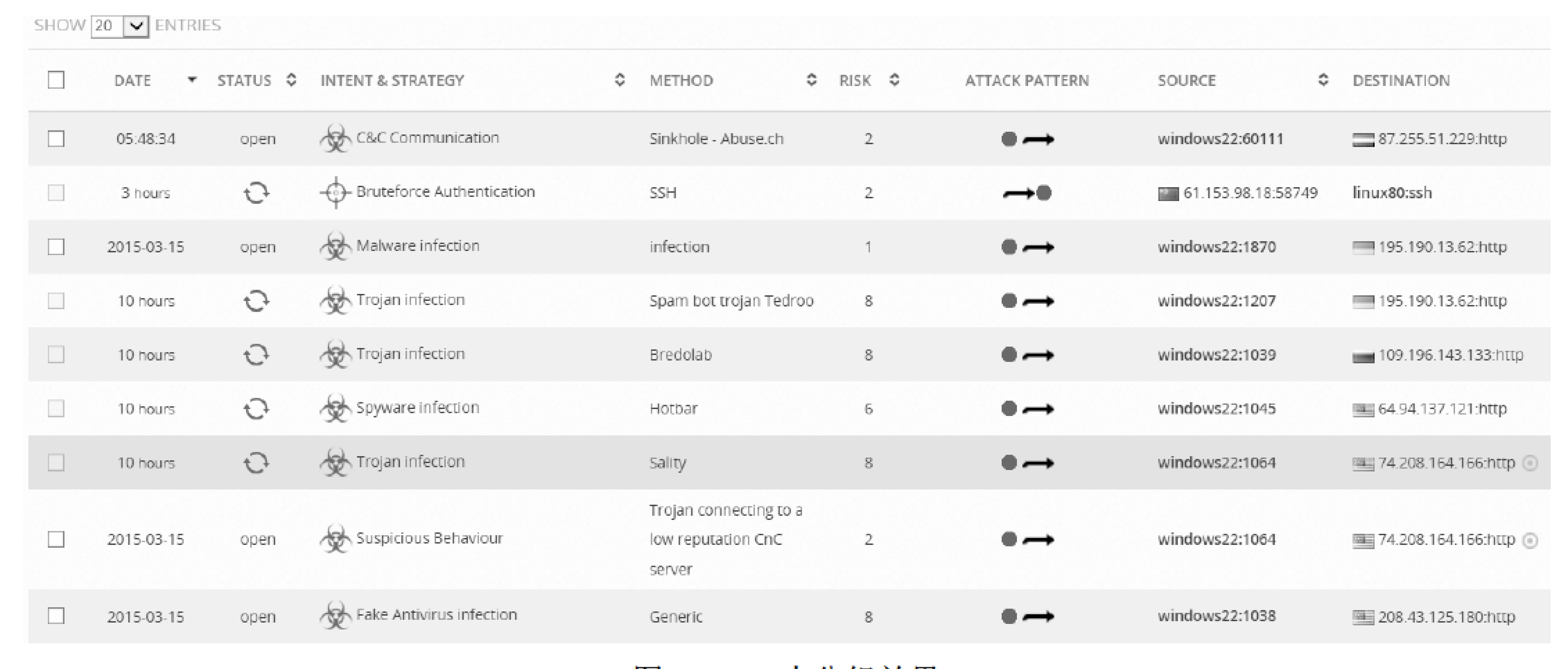


图 2-113　选择不同的传感器

（3）报警组

报警组的作用是形成事件报警的集合。报警组是 SIEM 控制台提供的有用功能，它允许报警分类显示出来，用于集中多种报警。可以把系统中一些试探性攻击、零碎的问题报警集中在一起。通过这个报警组功能，可以很快搜索到黑客攻击信息。

如图 2-114、图 2-115 所示，我们给出了未分组和分组后的效果比较。



SHOW 20 ENTRIES

DATE	STATUS	INTENT & STRATEGY	METHOD	RISK	ATTACK PATTERN	SOURCE	DESTINATION
05:48:34	open	C&C Communication	Sinkhole - Abuse.ch	2		windows22:60111	87.255.51.229:http
3 hours		Bruteforce Authentication	SSH	2		61.153.98.18:58749	linux80:ssh
2015-03-15	open	Malware infection	infection	1		windows22:1870	195.190.13.62:http
10 hours		Trojan infection	Spam bot trojan Tedroo	8		windows22:1207	195.190.13.62:http
10 hours		Trojan infection	Bredolab	8		windows22:1039	109.196.143.133:http
10 hours		Spyware infection	Hotbar	6		windows22:1045	64.94.137.121:http
10 hours		Trojan infection	Sality	8		windows22:1064	74.208.164.166:http
2015-03-15	open	Suspicious Behaviour	Trojan connecting to a low reputation CnC server	2		windows22:1064	74.208.164.166:http
2015-03-15	open	Fake Antivirus infection	Generic	8		windows22:1038	208.43.125.180:http

图 2-114　未分组效果

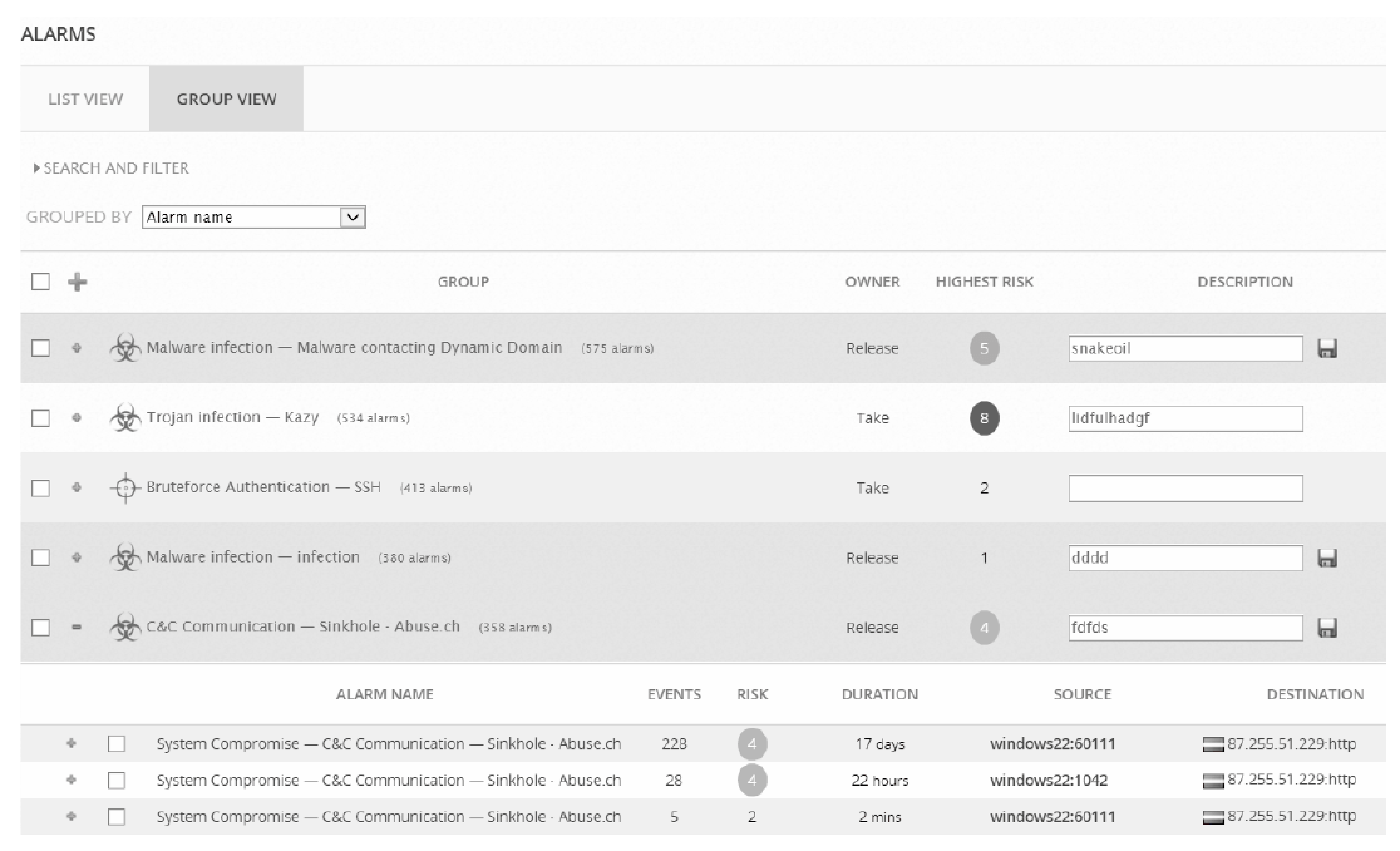


图 2-115　分组后的效果

（4）分类（Category）/子类

这种分类报警显示，在重新发现同一类攻击的早期行为方面特别有用。假如遭到针对特定服务的 Dos 攻击，那么通过这一查询，可知攻击者早就开始侦察踩点。即使是在 DoS 攻击中使用了欺骗 IP 都会被记录下来，那么使用了分类，将缩小查询范围，加快查询速度。Alarm 类下面包括 Attacks、Bruteforce、Dos、Malware、Scan 等之类，如图 2-116 所示。

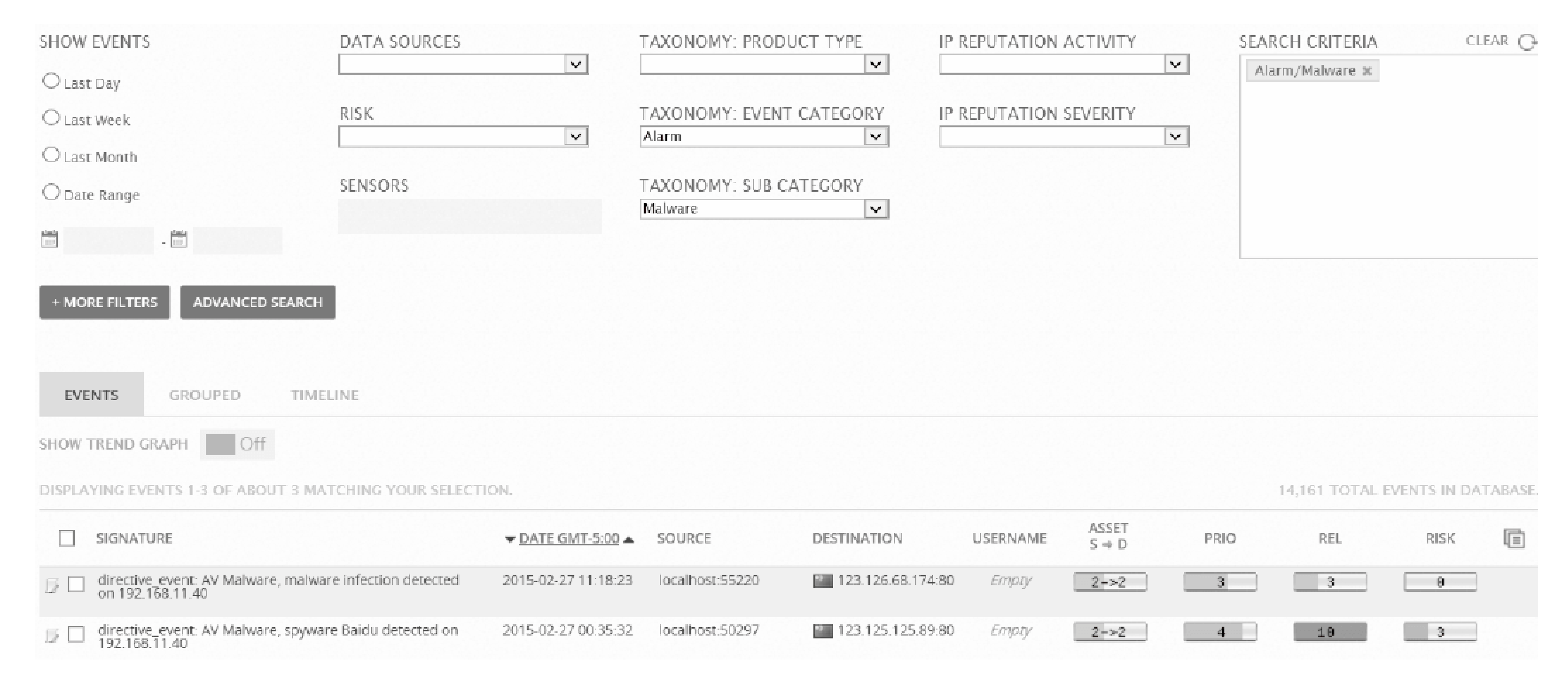


图 2-116　事件分类筛选显示

事件分类/子类的详细分类，我们可以在 Web UI 的 Configuration→Threat Intelligence→Taxonomy 菜单中查看，并能快速筛选内容。

（5）特征（signature）

特征码的作用是查询与特征值匹配的报警。在日常日志分析时，用特征匹配查询非常有用，尤其在分析 ShelloCode 攻击时非常方便。如图 2-117、图 2-118 所示。

图 2-117　Shellcode 攻击时的特征码

PAYLOAD
SHELLCODE
ANALYSIS

```
290 : 4C 49 4D 32 50 31 49 50 4B 43 4E 4A 4B 4E 47 32    LIM2P1IPKCNJKNG2
2a0 : 46 4D 4B 4E 47 32 46 4C 4D 43 4C 4D 43 4A 46 58    FMKNG2FLMCLMCJFX
2b0 : 4E 4B 4E 4B 4E 4B 45 38 43 42 4B 4E 48 33 45 46    NKNKNKE8CBKNH3EF
2c0 : 4B 4F 44 35 51 54 4B 4F 49 46 51 4B 50 57 50 52    KOD5QTKOIFQKPWPR
2d0 : 46 31 46 31 50 51 43 5A 43 31 46 31 50 51 50 55    F1F1PQCZC1F1PQPU
2e0 : 50 51 4B 4F 48 50 45 38 4E 4D 48 59 44 45 48 4E    PQKOHPE8NMHYDEHN
2f0 : 50 53 4B 4F 4E 36 42 4A 4B 4F 4B 4F 46 57 4B 4F    PSKON6BJKOKOFWKO
300 : 48 50 4C 4B 50 57 4B 4C 4C 43 49 54 42 44 4B 4F    HPLKPWKLLCITBDKO
310 : 49 46 50 52 4B 4F 48 50 42 48 4A 4F 48 4E 4D 30    IFPRKOHPBHJOHNM0
320 : 43 50 50 53 4B 4F 49 46 4B 4F 48 50 41 41 C3 AB    CPPSKOIFKOHPAA..
330 : 06 2C 49 E2 80 A1 21 02 10 48 C2 BB E2 80 A1 C3    .,I...!..H......
340 : B8 40 E2 84 A2 E2 80 98 41 6A 05 59 C3 99 C3 AE    .@......Aj.Y....
350 : C3 99 74 24 C3 B4 5B C2 81 73 13 2F 77 28 4B C6    ..t$..[..s./w(K.
360 : 92 C3 AB C3 BC C3 A2 C3 B4 C3 B6 E2 84 A2 C3 B1    ................
370 : 3F 0B C6 92 71 C3 8B C3 AE 7D C2 B8 C2 B5 C3 A2    ?...q....}......
380 : E2 80 B0 C3 A5 C2 B5 C3 A2 CB 86 C3 89 4B 5A 53    .............KZS
390 : 4E 55 52 46 45 42 47 54 4D 48 45 43 59 52 4C 53    NURFEBGTMHECYRLS
3a0 : 50 45 41 4F 4E 48 56 44 58 49 59 51 52 5A 58 44    PEAONHVDXIYQRZXD
3b0 : 4C 41 49 57 4E 59 7D 0D 0A                         LAIWNY}..
```

Snort rule Detection

File: emerging-shellcode.rules
Rule: alert tcp any any -> any any
msg: "ET SHELLCODE Rothenburg Shellcode"
flow: established
content: "|D9 74 24 F4 5B 81 73 13|"
content: "|83 EB FC E2 F4|"
distance: 0
reference: url,doc.emergingthreats.net/2009247
classtype: shellcode-detect
sid: 2009247
rev: 3

PCAP FILE: DOWNLOAD IN PCAP FORMAT

图 2-118　Shellcode 攻击的 Payload

（6）报警时间

该功能是为了查询特定时间内的报警，在各个时间段都会有大量的报警产生，可通过时间选项，进一步缩小查询范围。如图 2-119 所示。

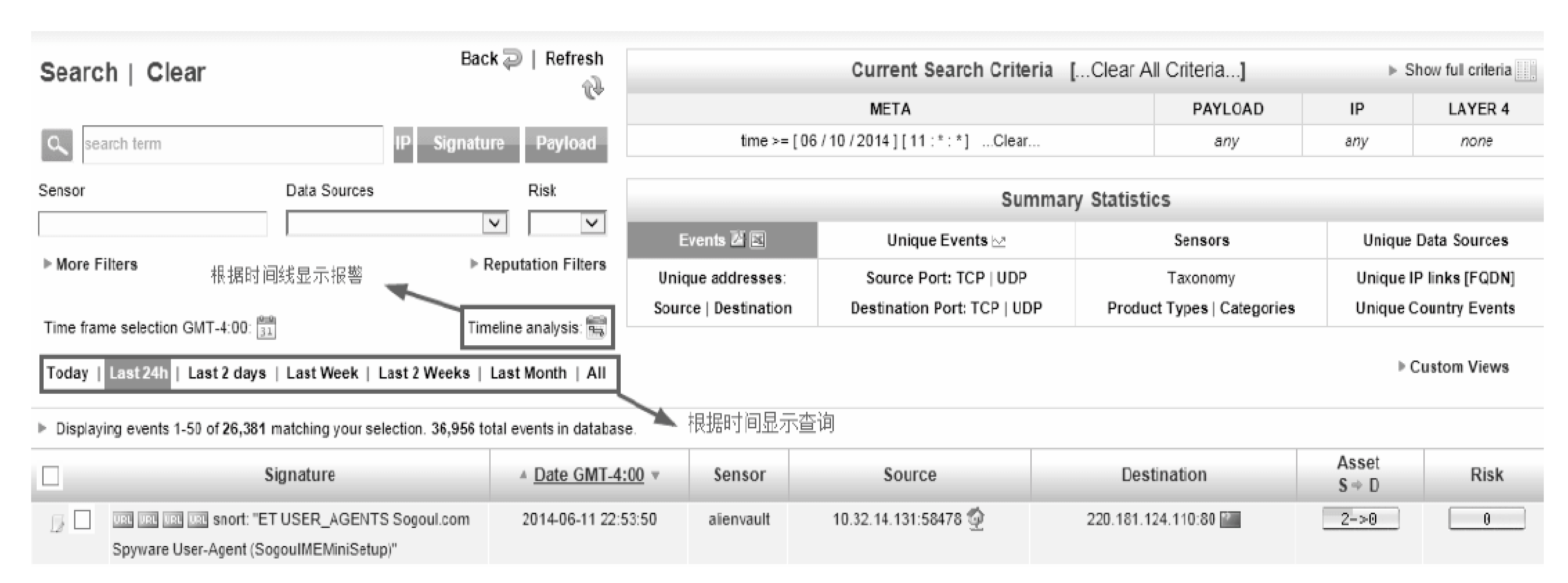


图 2-119　报警时间

在上图所示的界面上，单击 GMT+400 右侧的日历牌图标，可显示具体日期。

（7）IP 头数据

我们时常需要分析 IP 头数据，包含 IP 数据报的头部数据。从包头数据中我们可以获得哪些信息呢？

- 地址：可以鉴别出源地址、目标地址。
- 域数据：显示包括 IP 头部信息，包括服务类型、生存期、段 ID 标识、段偏移、报头校验及报头长度。读者熟悉这几部分的数据，异常的域数据往往是一次攻击或企图躲避 IDS 的征兆。

（8）传输层协议数据

该部分包含了 TCP 和 UDP 协议的信息，主要包含 TCP 端口、TCP 标志及 TCP 域数据（序列号、确认号、数据偏移量、保留位、窗口、校验以及紧急指针等）。

（9）有效负载（Payload）

有效载荷包含了应用程序要使用的数据，所有的报警中搜索存储在有效载荷中的数据串。这里的有效载荷可以包含从缓冲区溢出到各种木马的流量特征。通过 Payload 能快速发现隐藏的木马（前提是要知道木马的特征码）。我们先看这条报警的有效载荷，如图 2-120 所示。

Meta

ID #	Date GMT-4:00	Triggered Signature	Data Source Name	Data Source ID	Event Type ID
28 - 2901	2014-06-12 02:06:14	snort: "ET POLICY FTP Login Successful (non-anonymous)"	snort	1001	2003410

Source Address	Source Port	Destination Address	Destination Port	Protocol	Asset S → D	Priority	Reliability	Risk
10.32.14.123	21	10.32.14.131	64797	TCP	2->2	3	1	0->0

Sensor	Sensor Address	Interface
	10.32.14.133	eth0

Context Analysis: No action related to the context analysis

元数据

IP

Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
10.32.14.123	10.32.14.131	4	20	0	71	0	no	0	128	30152 = 0x75c8

Options: none

IP头数据

TCP

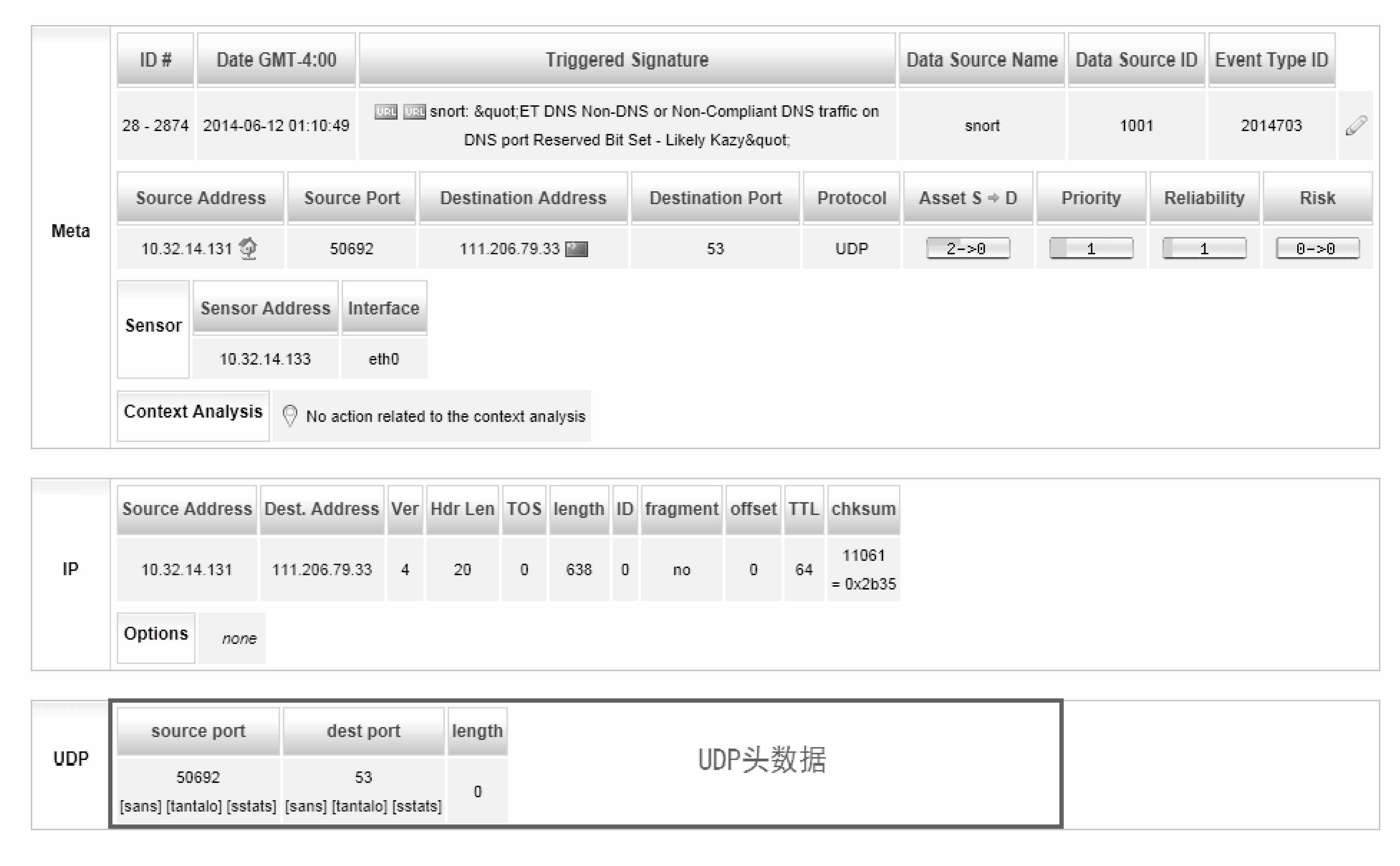
Source Port	Dest Port	R1	R0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
21 [sans] [tantalo] [sstats]	64797 [sans] [tantalo] [sstats]				X	X				2882468867	533465753	20	0	64212	0	27970 = 0x6d42

Options: none

TCP头数据

图 2-120　深度解析

再看另一条 SIEM 报警，这是 DNS 查询数据包，它包含了 UDP 头数据。如图 2-121 所示。



Meta

ID #	Date GMT-4:00	Triggered Signature	Data Source Name	Data Source ID	Event Type ID
28 - 2874	2014-06-12 01:10:49	snort: "ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port Reserved Bit Set - Likely Kazy"	snort	1001	2014703

Source Address	Source Port	Destination Address	Destination Port	Protocol	Asset S → D	Priority	Reliability	Risk
10.32.14.131	50692	111.206.79.33	53	UDP	2->0	1	1	0->0

Sensor	Sensor Address	Interface
	10.32.14.133	eth0

Context Analysis: No action related to the context analysis

IP

Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
10.32.14.131	111.206.79.33	4	20	0	638	0	no	0	64	11061 = 0x2b35

Options: none

UDP

source port	dest port	length
50692 [sans] [tantalo] [sstats]	53 [sans] [tantalo] [sstats]	0

UDP头数据

图 2-121　UDP 头数据

奇怪的 UDP 报文长度和失败的校验和，很可能暗示有攻击存在，例如一个长度为 0 的 UDP 包，系统认为是 DNS 查询而报警，其实有可能是一次 DOS 攻击。

（10）通过逻辑运算符查询

SIEM 控制台使用了一套扩展的逻辑运算符来建立查询。如图 2-122 所示。逻辑运算符定义出为建立查询而选择的标准元素之间的关系。这样做能进一步提炼想要查看的报警。

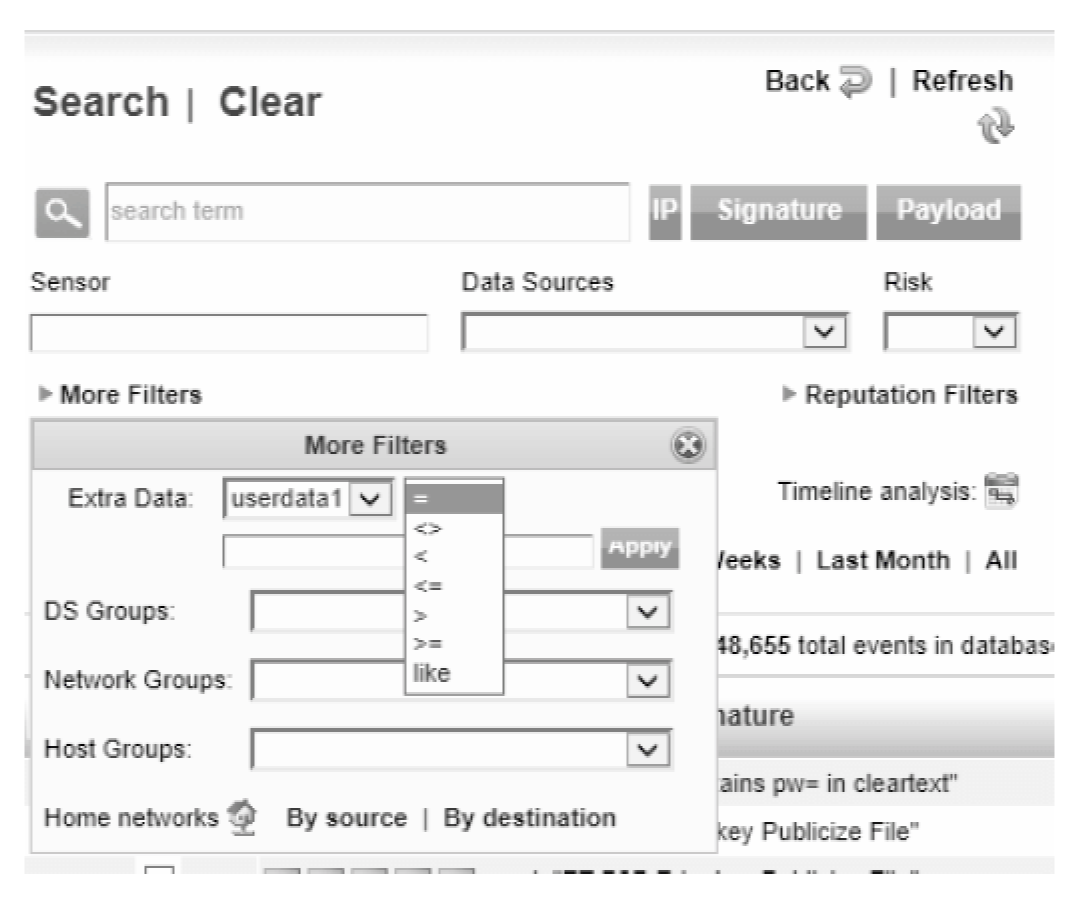


图 2-122　逻辑查询

这些运算符如下：

- =：等于。
- <>：不等于。
- <：小于。
- <=：小于等于。
- >：大于。
- >=：大于等于。
- Like：可以使用%作为通配符。

（11）过滤不同的 Sensor 输出

还可以在众多筛选条件中选取 Sensors 选项，来过滤由多个 Sensor 所输出的日志。如图 2-123 所示。

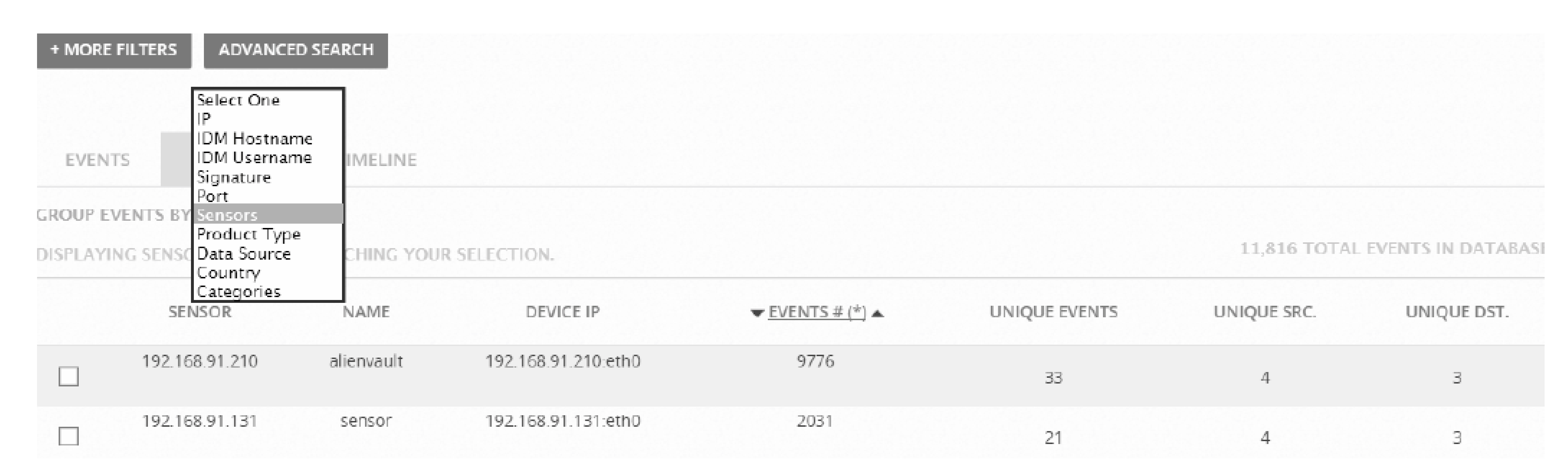


图 2-123　通过 Sensor 过滤日志

2.16.10　SIEM 警报中显示计算机名

采用 DNS 可以解析主机名，但有时候容易出问题，这里推荐采用静态解析的方式，为了让 OSSIM 能够顺利解析所监控的服务器发来的日志，我们需要配置 hosts 文件。我们知道 hosts 文件的作用相当于 DNS，Linux 系统在向 DNS 服务器发出域名解析请求之前，会查询/etc/hosts 文件，如果里面有相应的记录，就会使用 hosts 里面的记录。/etc/hosts 文件通常里面包含这一条记录：

```
127.0.0.1    localhost.localdomain   localhost
```

下面将所监控的所有服务器 IP 和主机名称对应，每条记录对应一台主机。然后分发到各台服务器上。

这个可以通过修改/etc/hosts 文件实现，hosts 文件是用来把主机名字映射到 IP 地址的方法，这种方法比较简单。但这种映射只是本地映射，也就是说每台机器都是独立的。

修改/etc/hosts 文件，实例如下（显示结果如图 2-124 所示）：

```
# cat /etc/hosts
127.0.0.1        localhost
10.32.X.Y      alienvault.alienvaultalienvault
10.32.X.Z      win7
```

	DATE	STATUS	INTENT & STRATEGY	METHOD	RISK	ATTACK PATTERN	SOURCE	DESTINATION
☐	2014-08-06	open	Malware infection	infection	2	●→	win7:60358	180.149.156.142:http
☐	2014-08-06	open	Software Update	Linux Package Manager	1	→	10.32.1[redacted]:46047	206.12.19.126:http
☐	2014-08-06	open	Malware infection	infection	2	●→	win7:56621	106.120.151.53:http
☐	2014-08-06	open	Software Update	Linux Package Manager	1	●→	alienvault:50374	70.38.37.7:http

图 2-124　SIEM 警报中以计算机名称显示

2.16.11　SIEM 事件保存期限

即使你有再大的硬盘，经过一段时间，也会被 Sensor 不断传输过来的各种告警所填满，而且关联分析引擎分析安全事件的总数也有个上限 所以新版 OSSIM USM 系统设定了以下限制：

- 在系统磁盘中保存 30 个备份文件（位置在/var/lib/ossim/backup/）。
- 在线可查询的安全事件保存 90 天（数值为 0，代表不备份）。
- 在数据库中保存 4000 万条记录（数值为 0 代表无限）。
- 备份时间为 01:00。
- Alarm 告警永久保存（也可设置生命期）。

具体参数我们可以到 Configuration→administration→Main 菜单下的 backup 选项中查看，如图 2-125 所示。

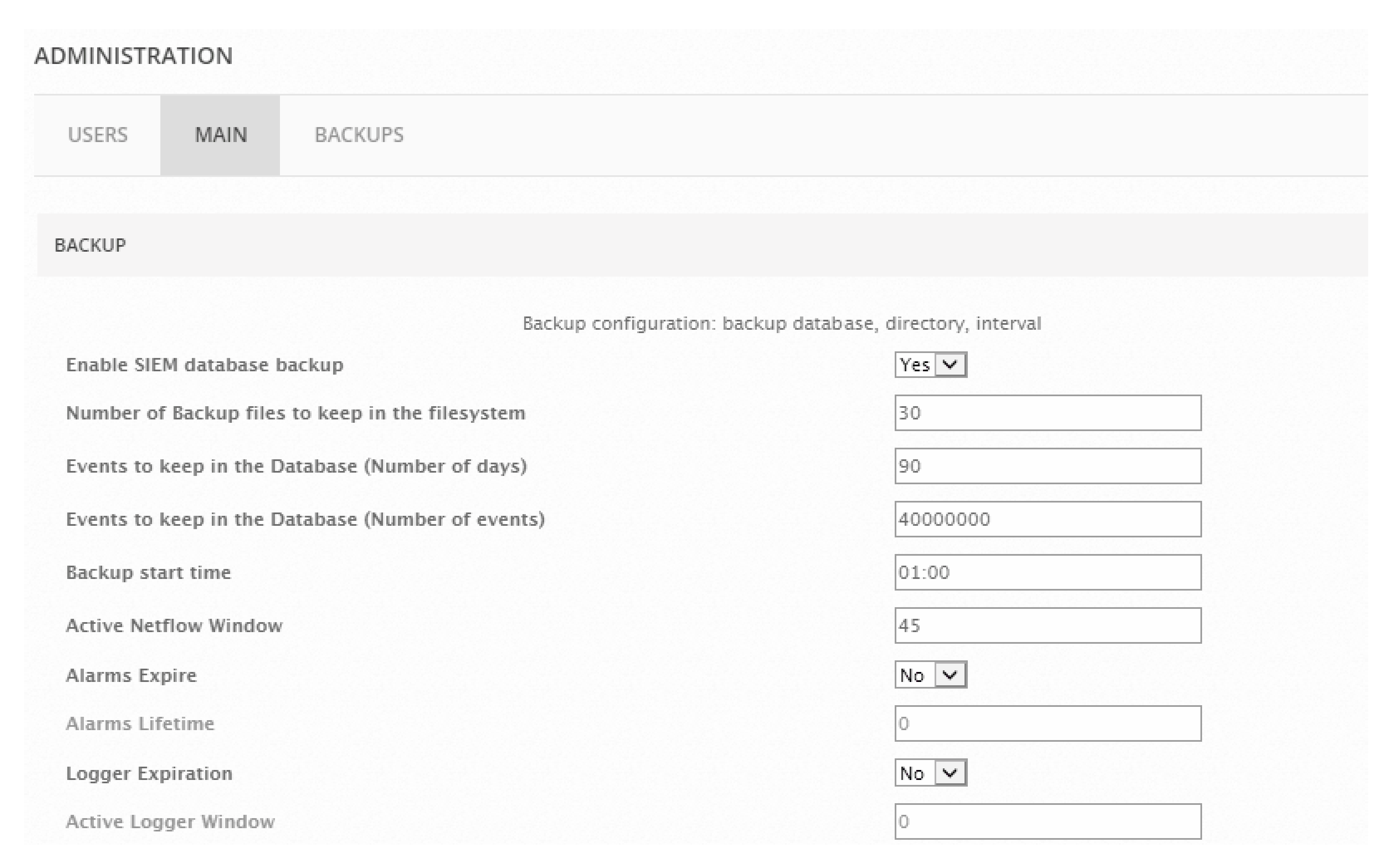


图 2-125　配置备份选项

2.16.12　SIEM 数据源与插件的关系

通过 Sensor 配置界面中启用的插件，能直观地反映在仪表盘和 SIEM 控制台上，如图 2-126 所示，图中以 aruba 插件为例说明（前提是已有该设备），首先启用 aruba 插件，应用生效后，便能在仪表盘中显示事件的多少。用户还可以在 SIEM 控制台上筛选 aruba 数据源所产生的所有事件。

图 2-126　数据源与插件的关系

2.16.13 SIEM 日志显示中出现 0.0.0.0 地址的含义

根据 RFC 文档描述，0.0.0.0/32 可以用作本机的源地址，作用是帮助路由器发送路由表中无法查询的包。如果设置了全零网络的路由，路由表中无法查询的包都将送到全零网络的路由中去。还记得吗，在路由器配置中可用 0.0.0.0/0 表示默认路由，作用是帮助路由器发送路由表中无法查询的包。严格说来 0.0.0.0 表示所有未知的主机和目的网络。这里的“未知”是指在本机的路由表里没有特定条目指明如何到达。

如果输入：

```
# netstat -anp | grep LISTEN | grep -v LISTENING
```

查看这条命令显示结果就比较好理解了。而在 OSSIM 系统的 SIEM 日志中查看到 0.0.0.0，它表示没有对应的 IP 与该日志相关联。

有时候在 SIEM 的 Web UI 下查看到 src ip 和 dst ip 也为 0.0.0.0，这是因为这些日志不涉及网络连接，为了填充这个字段，所以全部为 0，即没有源和目的 IP。还有种情况比较特别，当 OSSIM 主机解析失败时也会标记全 0 的地址，这时可以通过修改/etc/hosts 的方法手工逐条加入即可解决。另一种情况，当出现 ossec-syschek、ossec 本地文件系统进行检查时，目标地址及端口会被 0 填充。如图 2-127 所示。

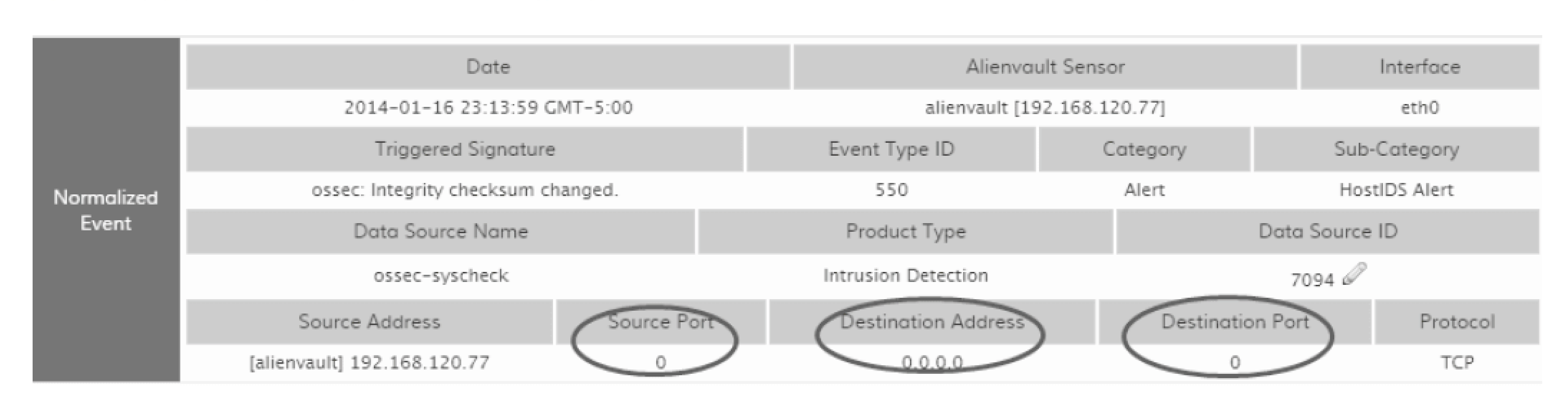


图 2-127　目标地址和目标端口为全 0

当遇到 Anomalies 流量时，异常流量通常是 Hacker 用拆包（通过数据包碎片传输）方法进行攻击防火墙等网络设备，这种攻击被称为 Anomalies，所以这种包，抓到后分析发现它的源地址、目标地址不全或没有。

2.16.14 无法显示 SIEM 安全事件时处理方法

当用户非法终止 alienvault-update 升级过程的数据库升级时，以及出现非法关机操作时，在 Web UI 中的 SIEM 控制台上，很有可能会出现下面的提示：

```
No events matching your search criteria have ben found.Try fewer conditions.
```

应该避免上述两种情况的发生，如果一旦遇到这样的提示，可以采用下面的方式解决。

（1）通过一条命令尝试修复：

```
#ossim-reconfig
```

（2）OSSIM 终端控制台重置 SIEM 数据库：

```
#ossim-setup        \\启动 OSSIM 终端控制台
```

以 OSSIM 4 为例，依次选择菜单 Maintenance&Troubleshooting→Maintain Database→Reset SIEM database。这时系统会启用脚本，注意先尝试修复数据库，实在不行再进行清理 SIEM 数据库。在重置 SIEM 数据库完成之后，首页仪表板上各项参数因缺少数据而无法显示，这时待检测数据积累到一定程度之后又可以重新显示。

2.16.15　SIEM 数据库恢复

重置 SIEM 数据库在终端下操作很容易，如果需要恢复到某一天的 SIEM 数据，则按图 2-128 所示提示操作。OSSIM USM 系统自动备份的环境数据库以天为单位，保存路径为/var/lib/ossim/backup，自动备份日志记录在/var/log/alienvault/api/backup-notifications.log 文件中。当用户需要恢复某天的数据，选中相应日期，例如在图中显示的日期“17-04-2015”，然后单击“Restore”按钮，经过数分钟后恢复完成。

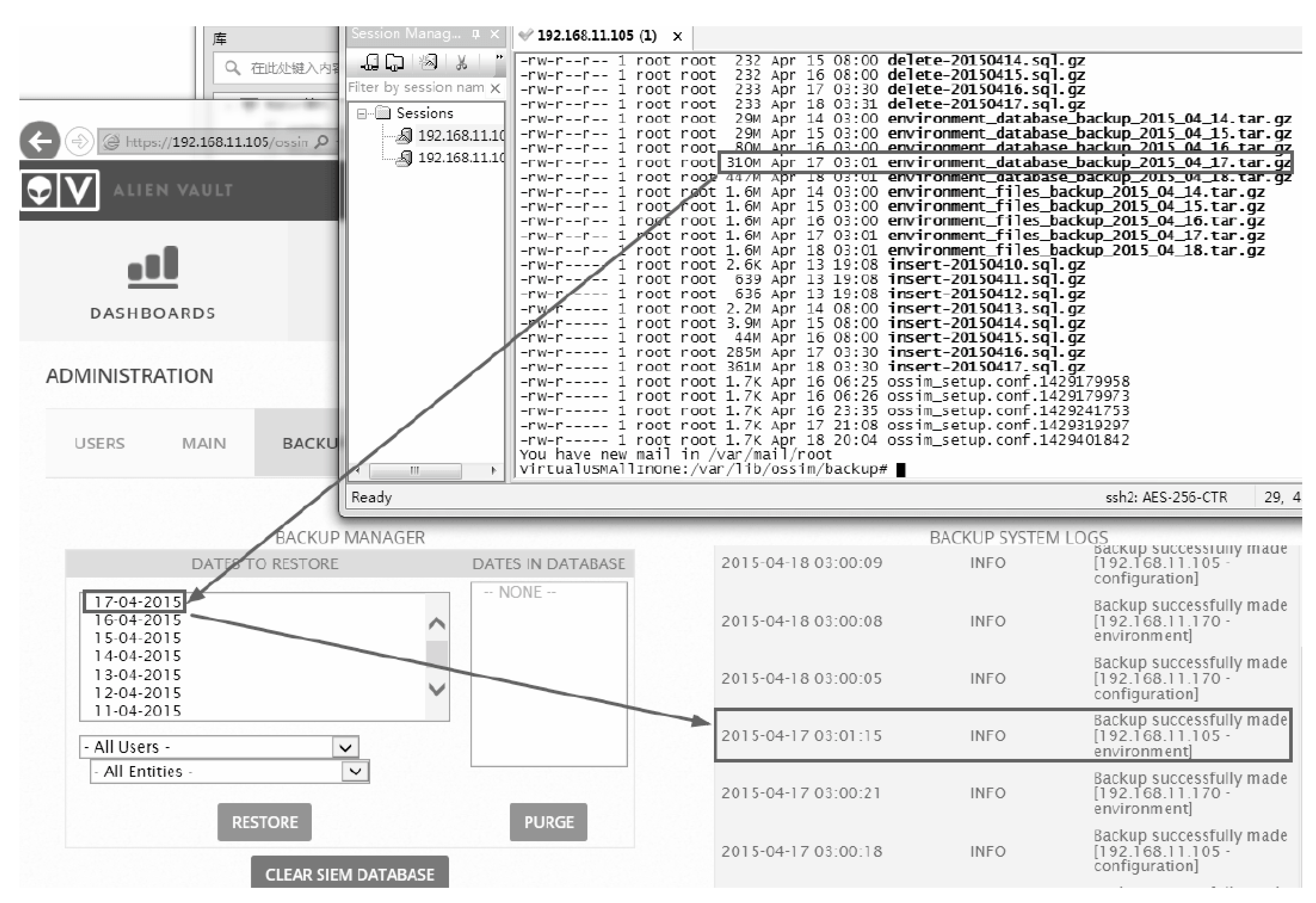


图 2-128　恢复 SIEM

2.16.16　EPS 的含义

EPS（Events Per Second）是系统每秒能处理的事件数量，这一数值反映了系统收集、处

理、整合事件的能力，同时反映了处理过的事件写入数据库的能力，假设系统平均 EPS 为 500，每天的事件数量为 4320 万条，那么从一个月的数据中找出符合条件的事件，对数据库将是个挑战。

在大数据时代，安全需主动，当发生事件数据增长超出预期峰值限制时，分析人员能否确定这种事件量增长是否由主动攻击引起的，这一点至关重要。OSSIM 系统不仅能够处理这些增长的情景，SIEM 将把每秒事件量（EPS）在最关键的时刻通知安全团队，以访问它们的主要态势感知工具，但是在免费版 OSSIM 系统中 EPS 为 500，企业版 EPS 的值可以达到 3000+。不过，这个值需要硬件配置足够（Xeon E5 8Core CPU，物理内存 32GB，完成关联分析 5000EPS 的任务）。每秒事件数量与 EPS 变化趋式如图 2-129 所示。EPS 变化趋势查看方法是：Web UI 中通过 Configuration→Deplayment→AlienVault center 选择节点名下的 OSSIM 主机，在 AlienVault 状况的右下角“VIEW TREND”中查看。

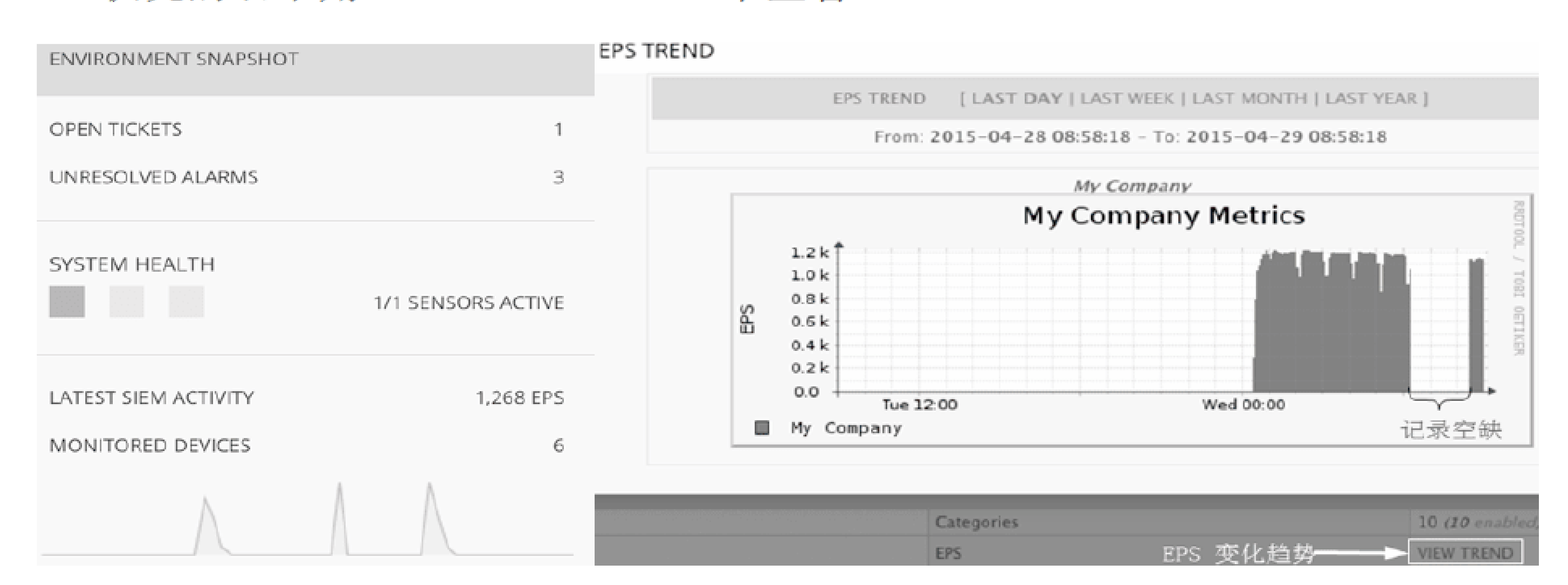


图 2-129　每秒事件数量 EPS

就国内 SIEM 产品而言，没有统一标准，目前还处于各自为战的状态。SIEM 系统每天存储几百万事件，而发现的真实故障问题可能也就几个，从用户角度理解希望 EPS 越大，证明系统处理能力越强，但从问题的发生→传输→接收→归一化处理→关联分析→告警→入库这个流程上看，EPS 越大系统（包括 CPU、Disk）承载的负荷就越大，所以用户在使用 OSSIM 这样的 SIEM 产品时，EPS 的大小可能每个用户都不同，这取决于系统架构，特别是数据库以及消息处理系统的配置。

还有一点需要明白，EPS 代表了服务器中被分析处理的事件数量，而不是通过 syslog 接收日志的数量。EPS 变化趋式在 OSSIM USM Server 端，在 sensor 端无数据。

2.16.17　常见 OSSIM 安装/使用错误

下面笔者列举了一些常见的 OSSIM 安装错误，以便大家在今后安装中少走弯路。

（1）OSSIM Server 以及 Sensor 的非法关机。

禁止 OSSIM Server 以及 Sensor 的非法关机，这将有可能造成数据库损坏。图 2-130 列举某一 OSSIM 系统在非法关机后，重开机的画面，左边为启动系统画面，但长期停留在此画面，但按 F2 键进入命令行界面后发现右边的提示，这就是非法关机后果，最后进入 OSSIM 单用

户模式使用 fsck 几经周折修复文件系统后才恢复系统。

```
Synthesizing the initial hotplug events...done.
Waiting for /dev to be fully populated...done.
Setting parameters of disc: (none).
Setting preliminary keymap...done.
Activating swap...done.
Checking root file system...fsck from util-linux-ng 2.17.2
/dev/sda1: Superblock last mount time (Wed May 20 00:43:03 2015,
        now = Sun May  3 19:32:13 2015) is in the future.


/dev/sda1: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
        (i.e., without -a or -p options)
fsck died with exit status 4
failed (code 4).
An automatic file system check (fsck) of the root filesystem failed. A manual fs
ck must be performed, then the system restarted. The fsck should be performed in
 maintenance mode with the root filesystem mounted in read-only mode. ... failed
!
The root filesystem is currently mounted in read-only mode. A maintenance shell
will now be started. After performing system maintenance, press CONTROL-D to ter
minate the maintenance shell and restart the system. ... (warning).
Give root password for maintenance
(or type Control-D to continue):
Login incorrect.
Give root password for maintenance
(or type Control-D to continue):
Login incorrect.
```

图 2-130 非法关机后的启动界面

（2）禁止在硬件配置较低的机器上运行 OSSIM 系统。

前面也谈到过 OSSIM 内存消耗问题，但大家在测试中还很可能在 2GB 或者 4GB 内存的旧服务器上安装 OSSIM 4 的 64 位版本，这种配置肯定没法使用。笔者在大量实践经验的基础上总结得出，至少 16GB 内存的服务器，系统才能稳定运行。如果读者在实验时，没有大内存的机器，那么只能选择低版本的 32 位 OSSIM 系统。例如 2GB~4GB 内存的单 CPU 服务器能够安装 OSSIM 2.3 版。

（3）OSSIM 流量收集分析口在交换机上没有正确设置 SPAN。

这种也属于初学者常见的问题之一，在使用 OSSIM 做 IDS 分析时，一定要将所检测网段的全部流量镜像过来，方法之一是采用 SPAN。当然对于流量很大的情况，可以采用网络分流器（Net-TAP）这种硬件卡实现分流目的。

（4）安装时仅安装了 Sensor 组件。

初学者在初次安装 OSSIM 系统时只安装了 Sensor，导致无法使用。

（5）OSSIM 装好了系统，长期置之不理。

OSSIM 虽然是一套智能分析系统，但也需要每天对其进行必要的维护，建议指定专门的网络安全分析师，经常观察 OSSIM 搜集的情报，以便及时进行调整。

（6）错误选用 SELinux 安装。

SELinux 全称是 Security Enhanced Linux，它目的在于明确指明某个进程可以访问哪些资源（包括文件，网络端口等）。有些朋友考虑加固 OSSIM 系统，因为系统默认没有启用 SELinux，所以想手工安装 SELinux。

```
#apt-get install selinux-basics
#apt-get install selinux-utils
#apt-get install setools
#selinux-activate
```

从安全角度考虑，这种想法没有错，但是 Alienvault 公司并没有为 OSSIM 组件做这方面的设置，冒然启用了 SELinux，后果很严重。SeLinux 不但改变了文件权限，而且还禁止了 so 库的调用，所以整个 OSSIM 系统无法启动。

（7）Server 和 Sensor 的角色能通过软件添加删除而互换？

不能，除了重装没有其他方法。

（8）启用过多插件。

对于新手常常喜欢将一些自己认识的插件全加载到系统中（尤其是混合式安装的 OSSIM 要承担更多压力），要知道插件内主要是正则表达式，当它在处理事件时是要消耗内存、磁盘空间和 CPU 等资源，加载越多消耗越大。当 Sensor 将事件归一化处理完之后，还需传给后续关联引擎多次分析，这样进一步加大系统负载，所以并不是添加越多越好。

（9）能否将 OSSIM 当成堡垒机使用？

不能。

（10）OSSIM 错误部署方式。

不要用 OSSIM 系统收集负载均衡服务器日志。每台负载均衡服务器每天会产生数百 GB 的访问日志，全网负载均衡服务器的日志量几十 TB，即使是通过 Gzip 压缩也有 3~4TB，这么大的负载会将 OSSIM 系统压垮。

（11）反复调整系统的时间/时区。

一定要设置好时区、时间，一旦调整好，系统运行一定阶段后，禁止再修改这个时间。否则 SIEM、日志会发生故障。

2.17 可视化网络攻击报警 Alarm 分析

在 Syslog 的日志级别中定义了一种叫做 Alert（警报）的日志，出现 Alert 意味着须马上采取行动的事件。本节开始为大家介绍 Alarm 及其分析的技术。在 OSSIM 系统中，Alarm 的各种提示应该是安全分析师调查事件的关键点。

在 OSSIM 的 Web UI 界面中，图形化 Alarm 报警由关联指令经关联分析引擎而生成，根源来源于 Snort 和 Ossec 等数据源的报警。长期安全事件分析中发现，绝大多数安全事件冗余出自同源、同类型、同目的的安全事件，同类型、同目的的安全事件或同类型、同源的安全事件所占比例更大，但在 Alarm 可以通过实时聚合算法来实现去除冗余，将多条安全事件聚合生成一条新的报警。此时读者可能想到安全事件聚合后输出到哪儿？下一个环节就是关联分析，比如根据交叉关联、序列关联来分析聚合后的事件，这样效率就比分析原始日志高得多，还能避免 IDS 带来的误报。

2.17.1 报警事件的产生

报警事件（Alarm Events）由关联指令（Correlation Directives）所产生，在 Alarm 中展示的警告信息是通过关联规则从大量的事件中匹配而得出。在 OSSIM 4.6 以后的系统，采用了一种图形化展示 OSSIM 系统 Alarm 报警的模式，这便于管理员从大量安全事件中筛选最重要的部分。Alarm 事件产生过程如图 2-131 所示。

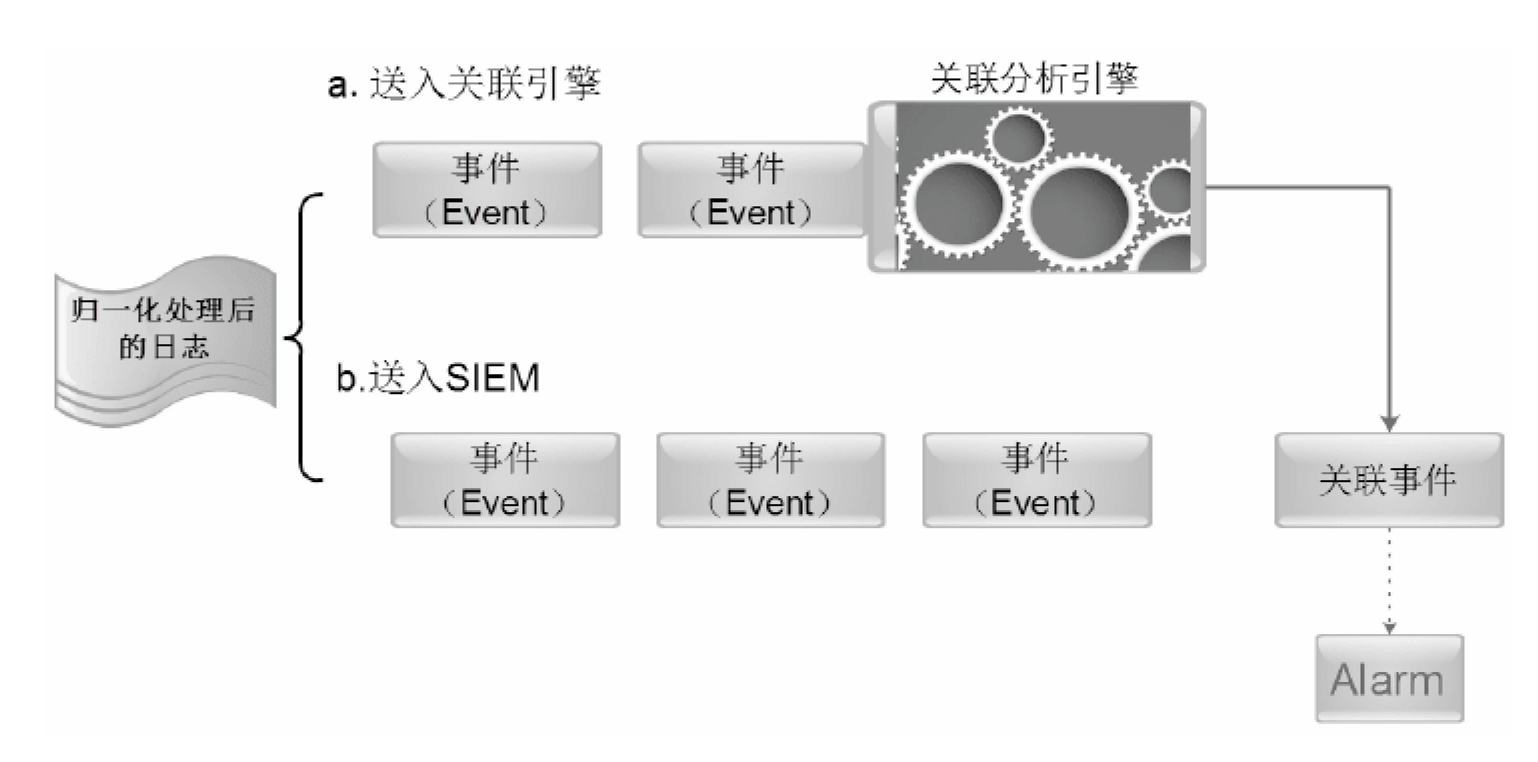


图 2-131 Alarm 事件产生过程

Alarm 生成步骤：

（1）日志收集到 OSSIM；

（2）将日志统一进行归一化处理后生成事件；

（3）将这些事件导入关联引擎；

（4）根据关联规则匹配出新的事件。

2.17.2 报警事件分类

在 OSSIM 的图形化报警显示中，由关联引擎处理后，生成的报警分为 5 种类型，产生图形化报警会比 SIEM 控制台里的安全事件有些延时。这几种类型的报警，读者可以到 Threat Intelligence→Directives 下查看关联指令，每种关联指令的下方都会标记指令对应的报警分类，总结如下：

- System Compromise System Compromise: 代表系统损害或破坏。图标采用感染性化学标记，对应策略文件为 alienvault-scan.xml 等。下面为该策略典型实例：

- Exploitation & Installation Exploitation & Installation: 代表漏洞利用与安装。采用警铃图标表示对应策略文件为 malware.xml 等。下面为该策略典型实例：

AV Web attack, SQLNinja successful attack against DST_IP
Exploitation & Installation WebServer Attack - SQL Injection, SQLNinja

- Delivery & Attack Delivery & Attack：代表攻击。采用瞄准镜图标表示对应策略文件为 alienvault-bruteforce.xml alienvault-attack 等。下面为该策略典型实例：

AV-FREE-FEED Web attack, SQL injection attacks detected against DST_IP
Delivery & Attack WebServer Attack - SQL Injection, Attack Pattern Detection

- Reconnaissance & Probing Reconnaissance & Probing：代表侦查与探测。采用雷达扫描图标表示对应策略文件为 alienvault-scan.xml 等。下面为该策略典型实例：

AV-FREE-FEED Network scan, SSH service discovery activity from SRC_IP
Reconnaissance & Probing Service discovery, SSH

- Environmental Awareness Environmental Awareness：代表环境意识。对应策略文件为 alienvault-policy.xml 等。下面为该策略典型实例：

AV-FREE-FEED Policy violation, BitTorrent P2P usage on SRC_IP
Environmental Awareness Desktop Software - P2P, BitTorrent

这五类报警在 Web UI 中显示在同一张图中，通过菜单 Analysis→Alarm 查看，如图 2-132 所示。从图中看出圆圈面积大小，直观地反映了某类事件数量的多少。

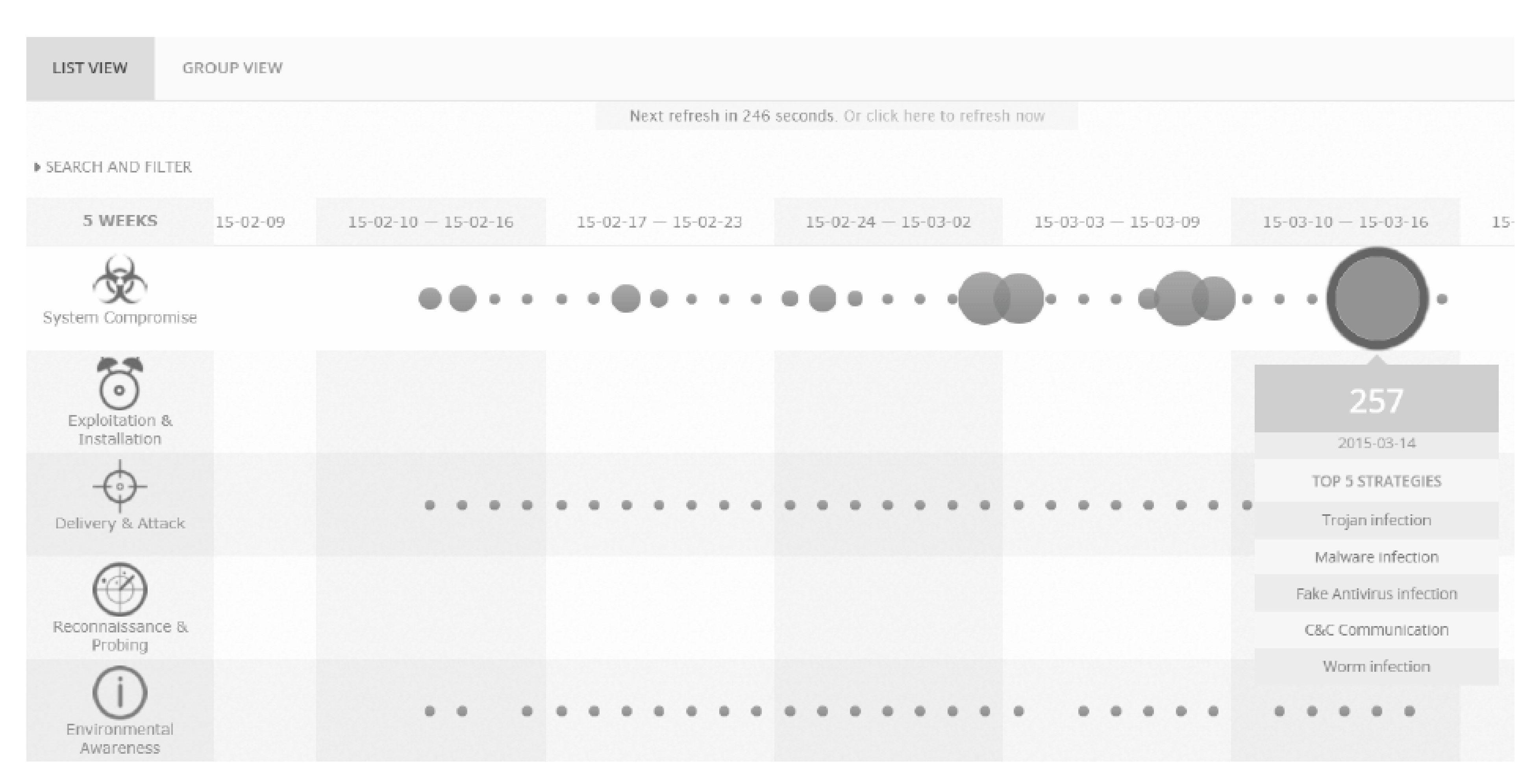


图 2-132　Alarm 种类

1. System Compromise

此类报警属于系统危害类安全事件，出现此类报警说明攻击者已通过网络或其他技术手段造成系统中信息被篡改、信息泄露与被窃取（未授权用户获取信息）。例如当系统感染蠕虫后，

会对系统造成不同程度的损害，在图 2-133 中显示了感染蠕虫时系统损害实例。它的模式是从源地址发往目的地址，系统给出了原地址和目的地址。这类报警放在首位，说明它是最重要的，最引人警觉的报警，往往这类报警的风险大于 3（属于高风险事件），大家应该首先关注这类报警。

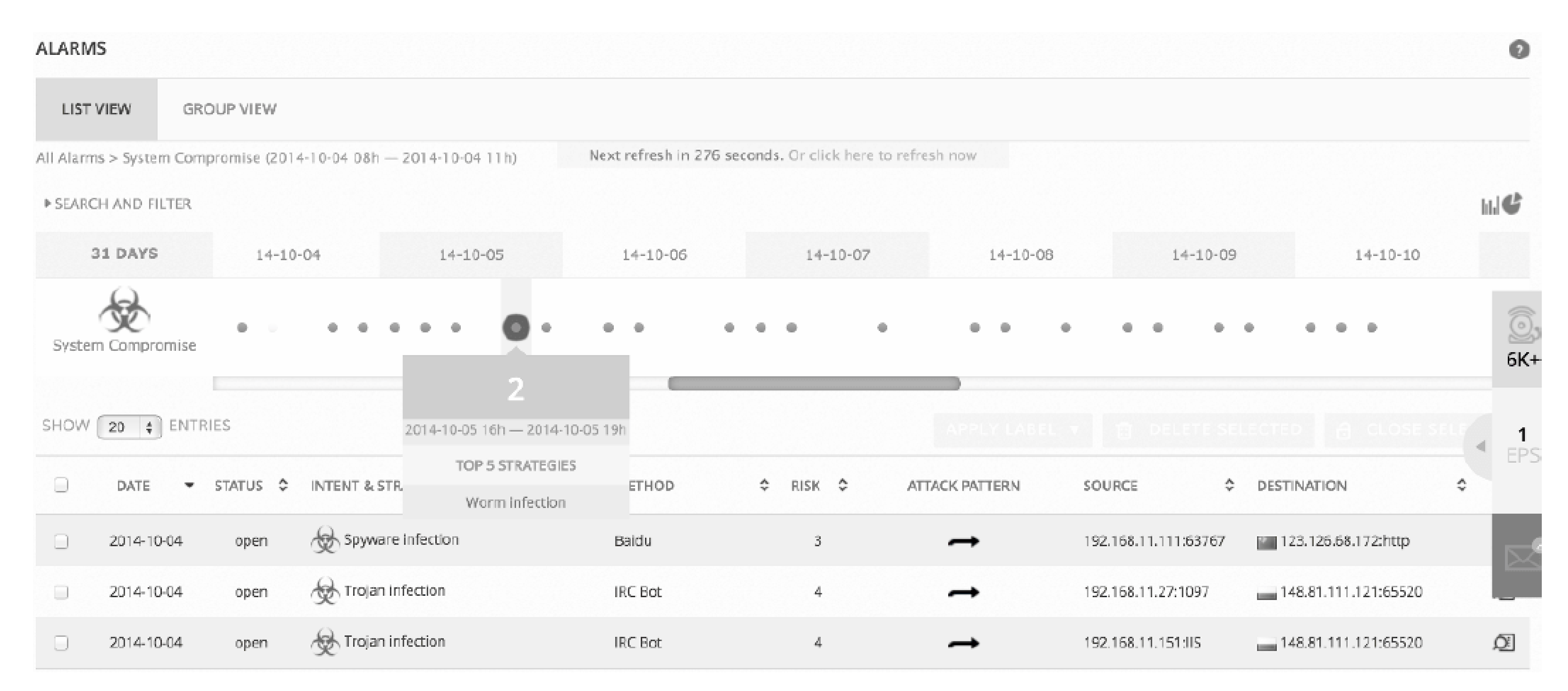


图 2-133　系统损害实例

对于隐藏性非常强的 C&C 攻击，通过这里的报警同样能显示出来，如图 2-134 所示。

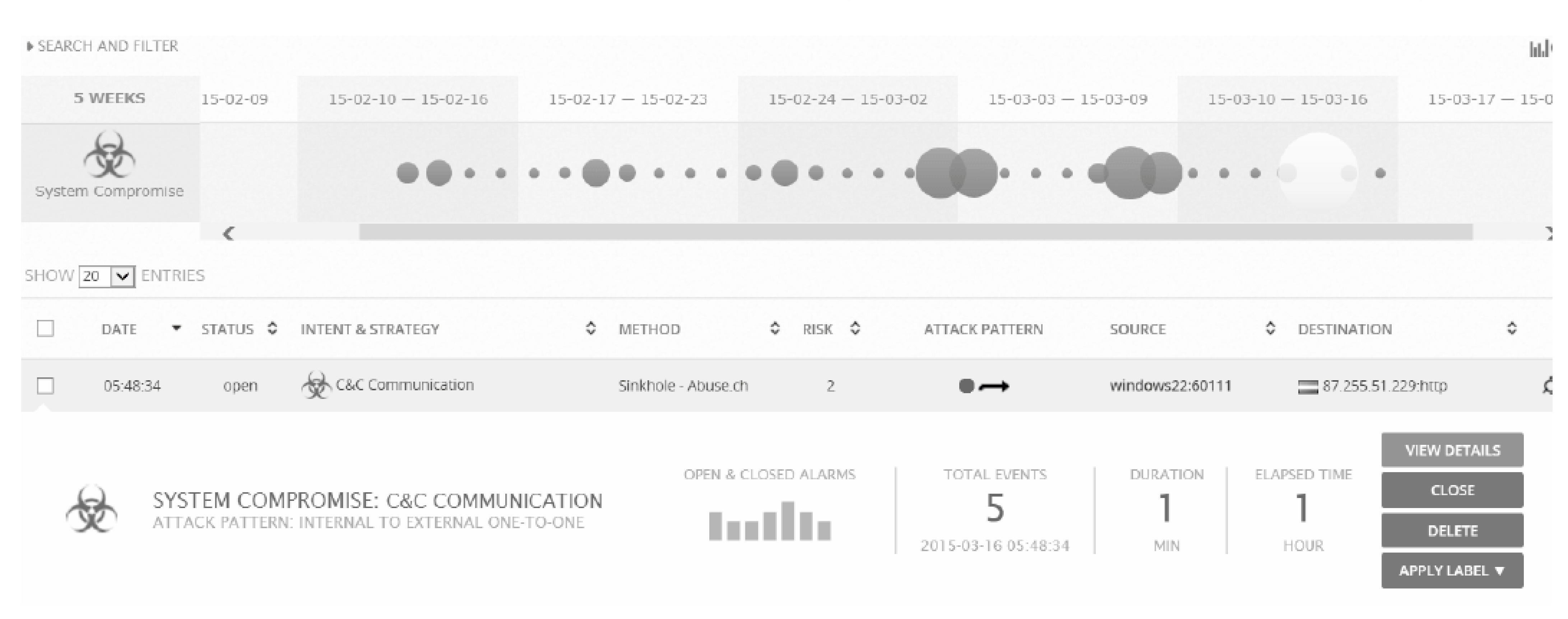


图 2-134　C&C 攻击危害检测

在事件特征码中，单击右键可以看到“Logs by Signature”等 3 个按钮，在原地址处单击右键会显示 Asset Detail 等 14 个和这条事件相关的功能菜单，大家可以自己逐个尝试。如图 2-135 所示。

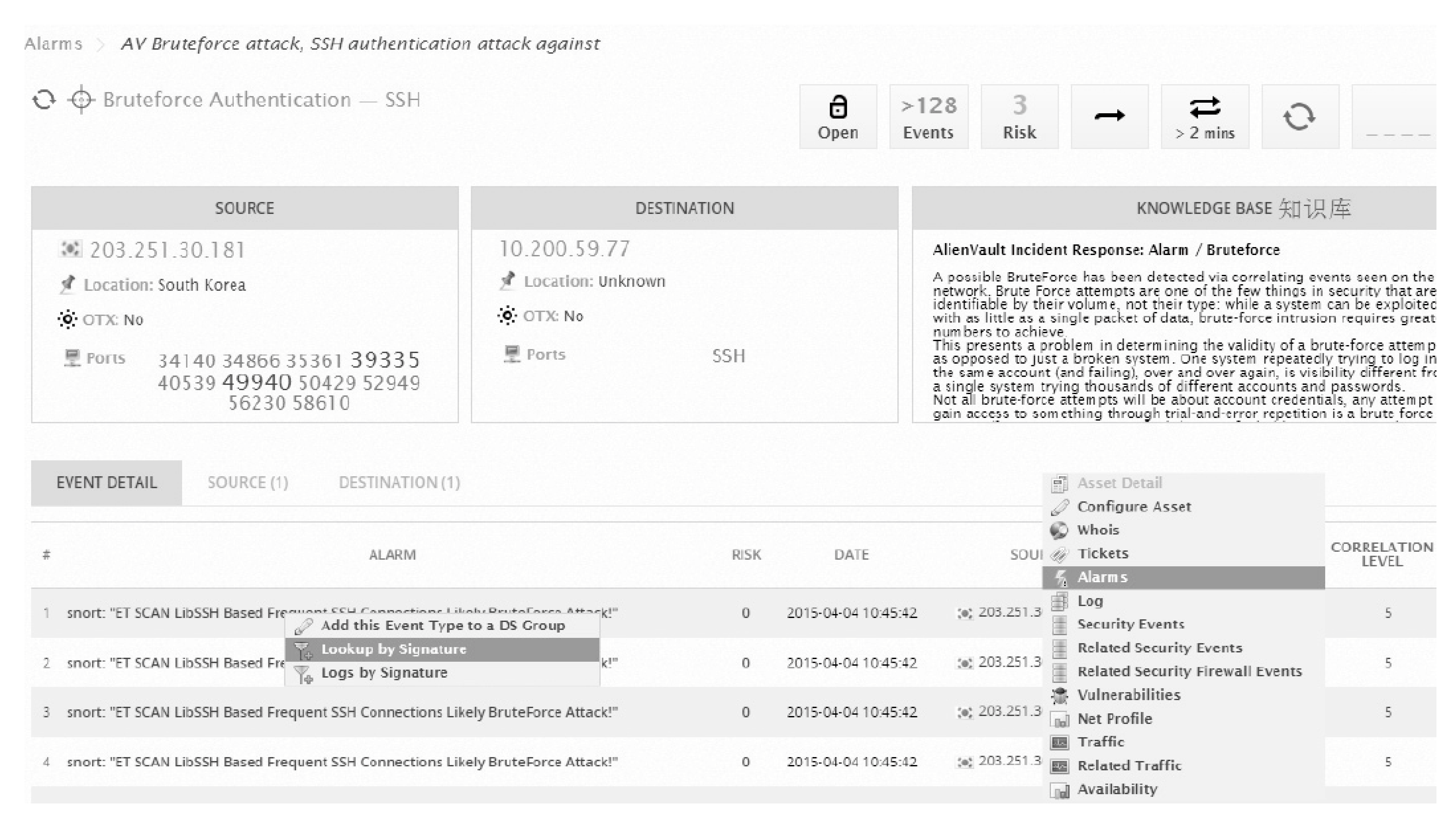


图 2-135　SSH 暴力破解时 Snort 报警

2. Exploitation&Installation

第二类重要的告警为 exploit（漏洞利用程序），它属于恶意代码类安全事件，攻击者蓄意制造、传播恶意代码，包括木马、网页挂马、跨站脚本 XSS 和僵尸软件，这通常意味着攻击者进行了渗透、提权等严重的攻击事件。在图 2-136 中的告警表示 Web 服务器遭受了 XSS 攻击。

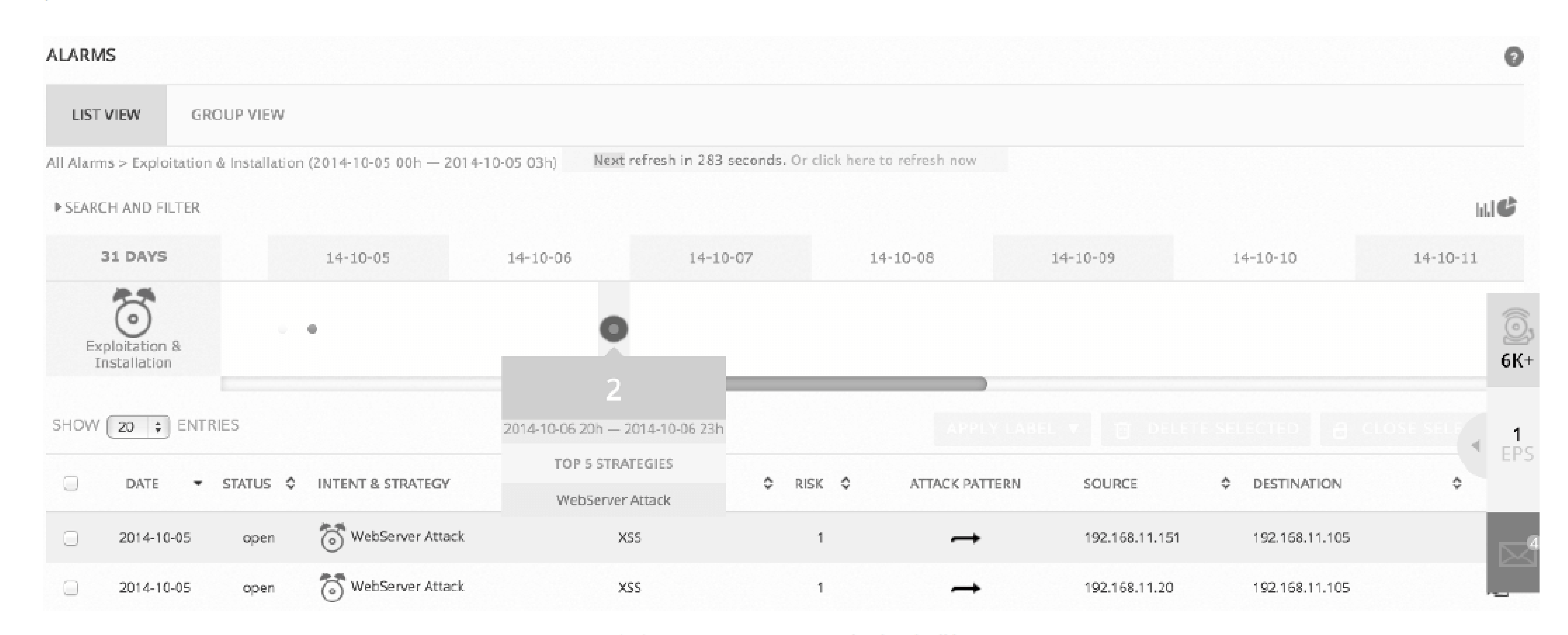


图 2-136　XSS 攻击告警

3. Delivery&Attack

Delivery 和 Attack 报警表示正在发生攻击行为，这属于入侵、攻击类安全事件，它利用系统配置缺陷、协议缺陷、程序缺陷，使用暴力攻击等手段对信息系统实施攻击，包括 Dos、猜测口令、域名劫持、SQL 注入等。如图 2-137 所示，表示系统捕获的暴力破解事件。

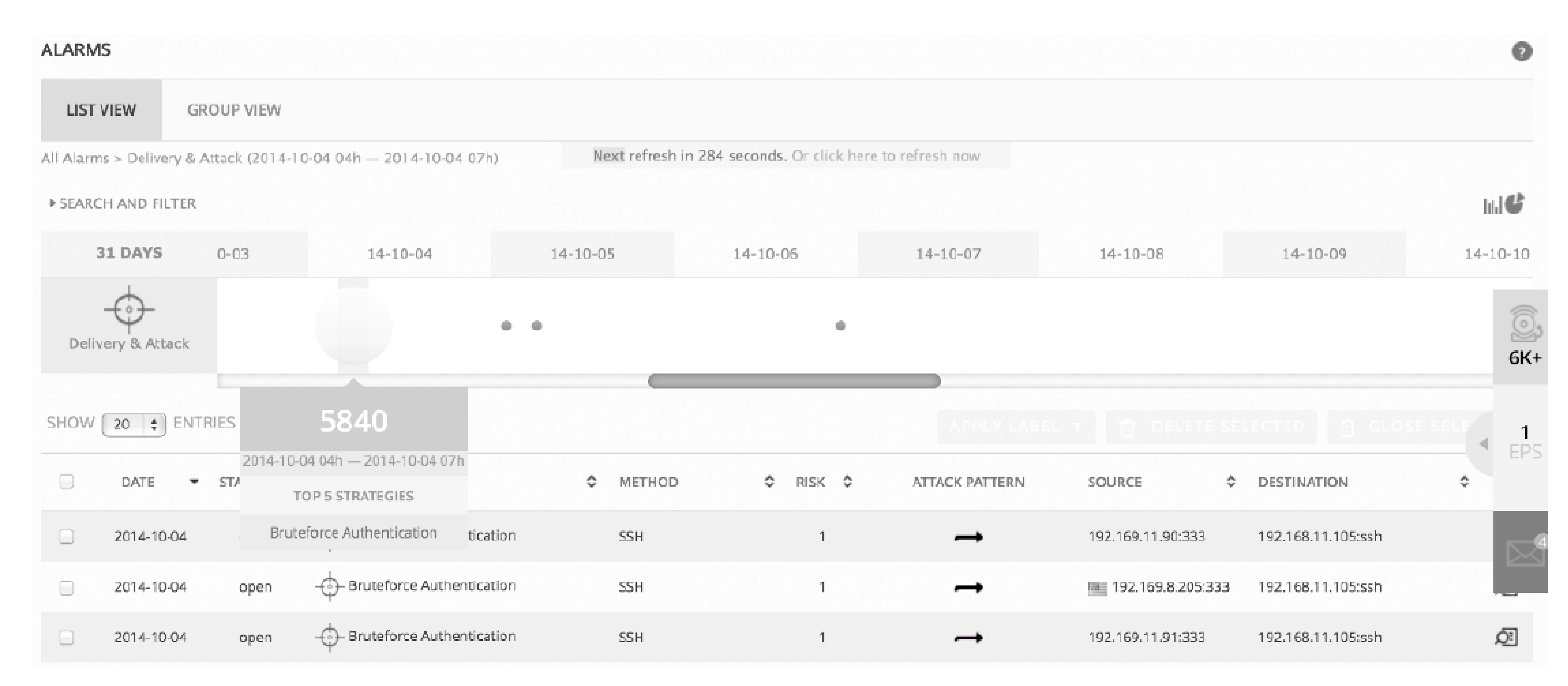


图 2-137　检测到攻击者对 SSH 服务进行暴力破解

4. Reconnaissance&Probing

这属于扫描探测类安全事件，它是用嗅探或模拟业务通信的方式获得系统及网络信息的各类事件，如目标存活信息、端口服务开放信息、操作系统指纹等。对入侵攻击来说，扫描探测是信息收集的主要手段，所以通过对各种扫描原理进行分析后，我们可以找到在攻击发生时数据流所具有的特征，而 OSSIM 的告警功能可以发现这种异常行为，图 2-138 展示了端口扫描（还包括系统服务扫描和通信协议扫描）行为。

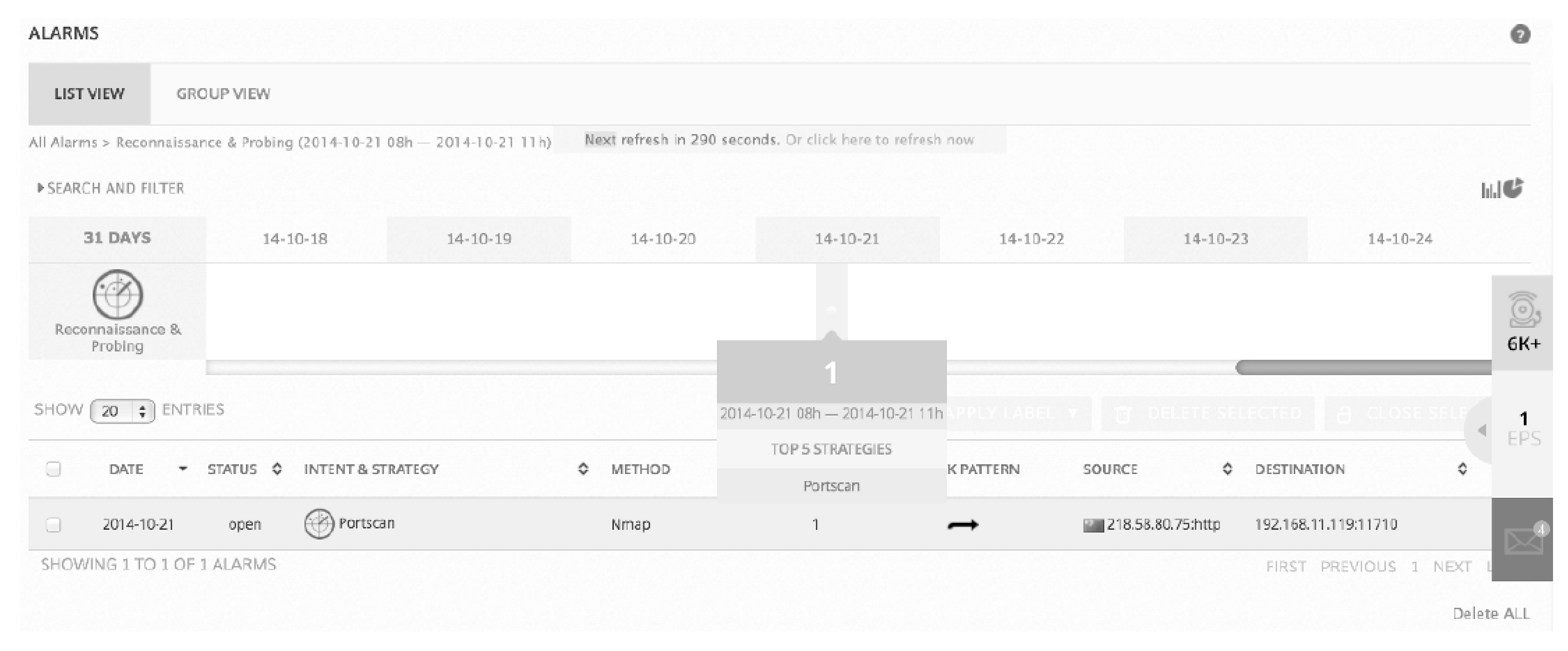


图 2-138　检测到端口扫描

5. Environmental Awareness

这类行为的告警优先级最低，通常是软件升级或者电驴 BT 等 P2P 类下载的报警，包含易受攻击的软件和可疑的通信，其 Risk 值一般≤1，如图 2-139 所示。这种报警有时却不容忽视，例如当某一台服务器，未经自己操作，然后在这里出现了升级的痕迹，那么管理员也应查明到底是你同事操作过，还是黑客已经接管了服务器。

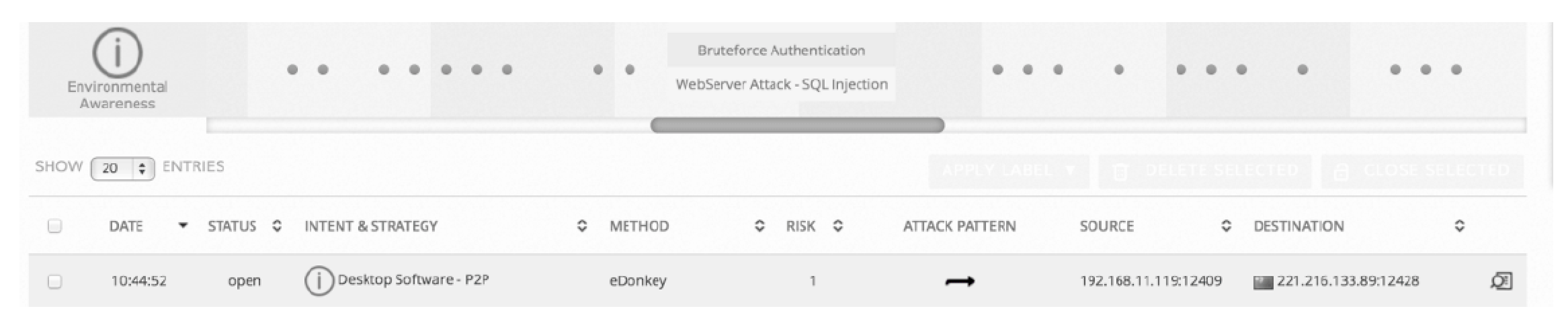


图 2-139　检测到 P2P 下载

2.17.3　五类报警数据包样本下载

OSSIM 将报警类型分为上一小节所说的五种类型，大家在工作中或实验中，一般不会全部测试出效果，也就是说，有些报警可能短时间出现，为了方便大家实验，这里给出了一个方便快捷且无害的方式，即通过下载特殊的封包的方式实现快速报警，这些封包通过蜜罐抓取，经特殊处理之后，保存为.pcap 格式。大家只需下载这些数据包，然后在 Sensor 所监控的网段重放，即可立即获得报警。下载地址为 http://chenguang.blog.51cto.com/ 。

2.17.4　报警分组

在 Alarm 下的告警列表中，还提供了另一种简明扼要的方式展现告警。大家可以尝试单击 Group View，告警分组，如图 2-140 所示，可以将告警分类显示出来。

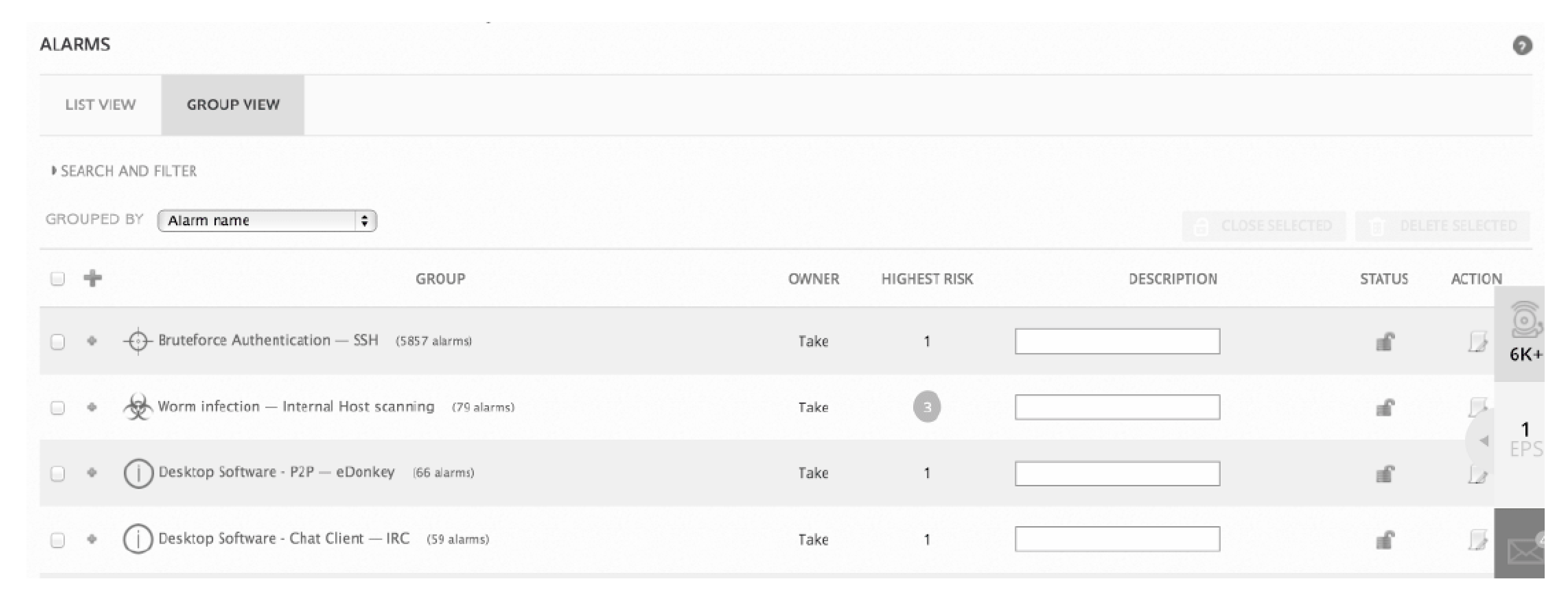


图 2-140　告警分组显示

这种分类的好处是只显示出有效告警。若想在分组中查看某一分组中某类告警的详细信息，只要单击对应事件即可，如图 2-141 所示。

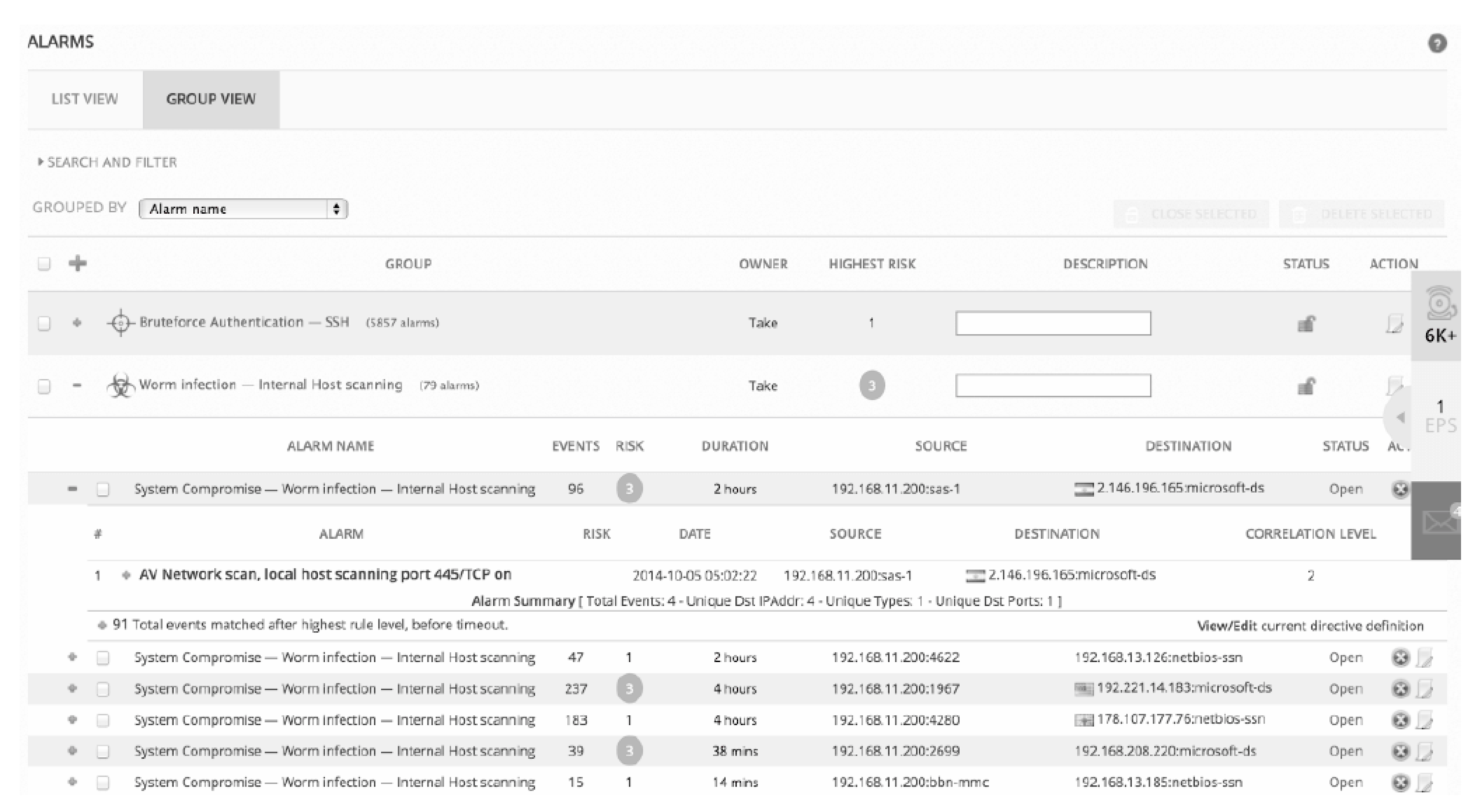


图 2-141　感染蠕虫的告警

下面仅以 System Compromise→Worm Infection →Internal Host scanning 告警为例，在系统危害中出现了内部主机的扫描行为，这类行为疑似为网络蠕虫的扫描，扫描主机漏洞往往是蠕虫传播的前提，蠕虫通过 ICMP 包、TCP SYN、FIN、RST 以及 ACK 包探测，且具有随机性。从图 2-142 中看出，系统定义出这类事件的风险值、扫描的持续时间、源地址、目的地址、源端口、目标端口以及关联等级，这种异常行为会立即被 OSSIM 发现。告警截图如图 2-142 所示。

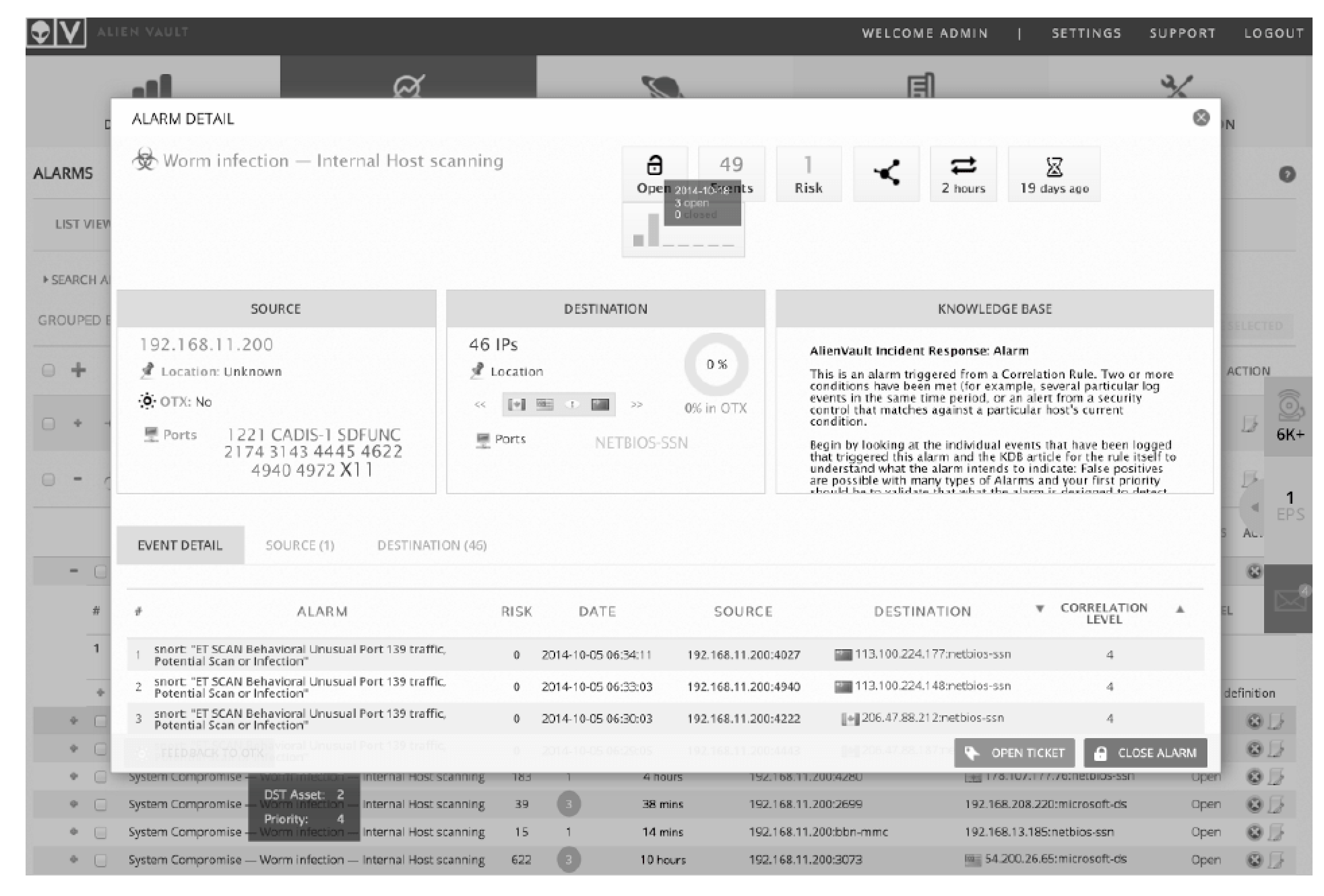


图 2-142　查看告警事件详情

从整体上发现某段时间内发生了多少种告警还不够，还需掌握告警的具体信息以及它的发展趋势、时间、背景知识库等信息，这些为我们诊断故障提供了重要依据。

2.17.5 识别告警真伪

谈到真、假告警的问题，就如同医院中病毒检测一样，因此，OSSIM 的 IDS 系统数据经过关联分析引擎检测会得出一个报告，那么我们不能完全按这个报告去判断，如果把正常的资源访问判断为入侵行为并加以告警，就是“误报（False Positive）”。如果没有发现某个入侵行为，没有对其进行响应，就是“漏报（False Negative）”。

安全策略定义得过于严格，就容易产生“误报”。例如某些只会针对 Windows 攻击才会成功的攻击，如果它的目标节点系统为 Linux，显然不能成功。某些只能针对 Windows 低版本才能成功的攻击，如果目标节点为 Window8/10 系统，这种攻击也不能成功。对于这类事件，同样会产生安全报警，但不具有危害，系统会自动标记为误报。

安全策略如果定义得过于宽松，就容易产生“漏报”。与误报相比，漏报其实更危险。IDS 想要防止欺骗，就要尽可能地模仿 TCP/IP 栈的实现。但是从效率和实现的复杂性考虑，IDS 并不能做到这一点。IDS 的实现总是在漏报和误报中徘徊，漏报率降低了，误报率就会提高，反之亦然。

网络中不断有新的威胁产生，在 Web UI 中 Analysis→Alarm 菜单可以将所有告警进行聚合并分类，同时允许管理员将聚合后的告警进行标识，如图 2-143 所示，选择“Vulerable software”，在右下角会出现“APPLY LABLE”标签，弹出两个选项分别是“Analysis in Progress（分析进展情况）”，用红色字体醒目标出，含义是继续分析该报警。另一种是“False Positive”表示误报系统，用蓝色字体醒目标出，表示可忽略的报警，如图 2-143 所示。

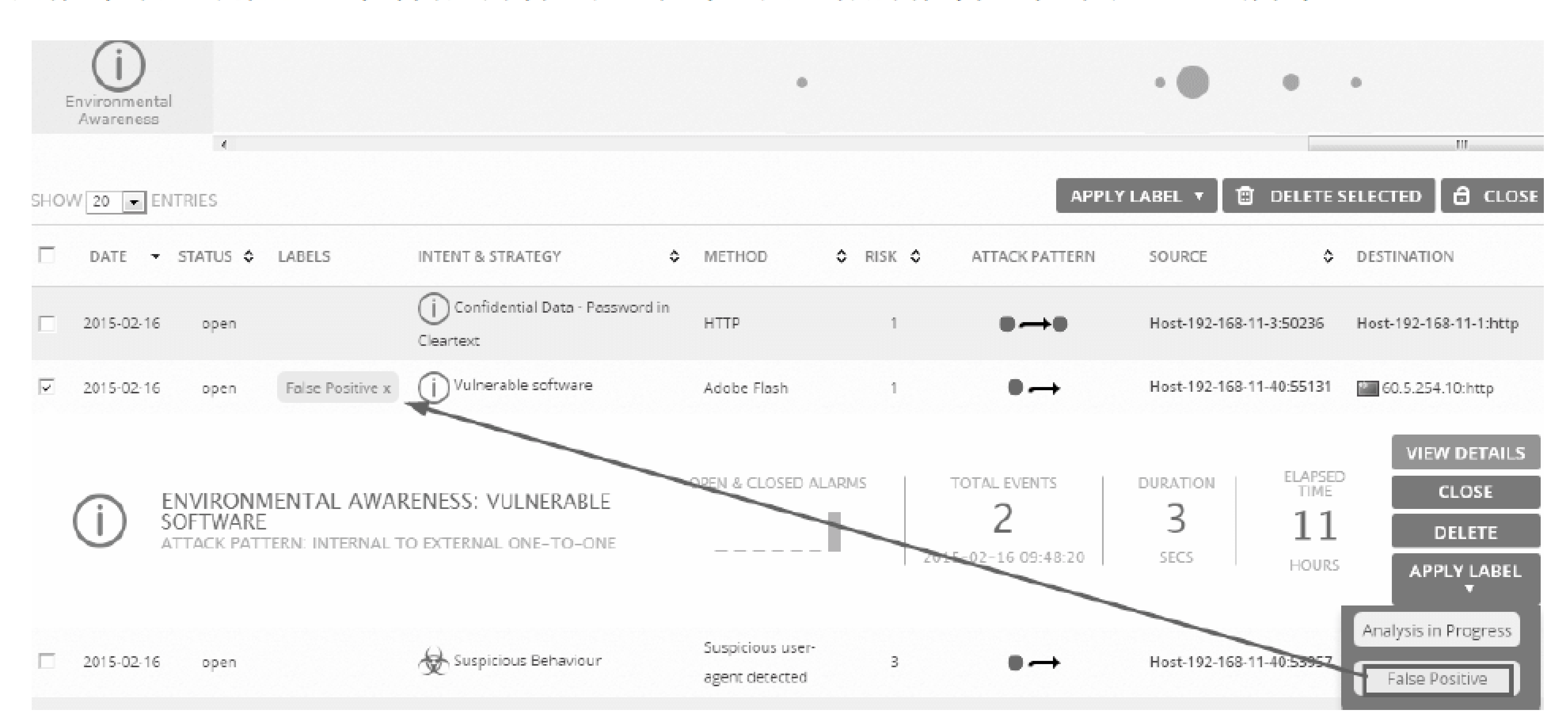


图 2-143 告警标识

2.17.6 触发 OSSIM 报警

当使用扫描器或渗透工具对 OSSIM Sensor 所监控网段的主机进行扫描和渗透，SIEM 控

制台会发出事件报警。对于扫描工具，大家可以采用 BT5 光盘中的工具（最新版本为 Kali-Linux），例如 nmap、Autoscan、fping3 等，渗透工具可采用 Metasploit。

BT5 系统中集成了 V4 版本的 Metasploit 渗透测试框架，在 Terminal 上执行 msfconsole 命令就可以直接进入 Metasploit 的命令行界面。BT5 系统中集成的 Metasploit 渗透测试框架不是最新版本，使用前首先进行更新。在 Terminal 中通过执行“msfupdate”命令对 Metasploit 渗透测试框架进行更新，如图 2-144 所示，更新之后便可以体验最新版本的 Metasploit。

```
root@kali:~# msfupdate
[*]
[*] Attempting to update the Metasploit Framework...
[*]

[*] Checking for updates
[*] Updating to version 4.11.0-2014122301-1kali0
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  metasploit metasploit-framework
2 upgraded, 0 newly installed, 0 to remove and 480 not upgraded.
Need to get 223 MB of archives.
After this operation, 125 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali/ kali/main metasploit-framework i386 4.11.0-2014
122301-1kali0 [55.9 MB]
24% [1 metasploit-framework 54.1 MB/55.9 MB 97%]             133 kB/s 21min 13s
```

图 2-144　升级数据库

我们在 Kali 下启动 Metasploit 图形界面，Armitage 是一款用 Java 编写的 Metasploit 图形界面工具，可用来针对存在的漏洞进行自动化测试（注意在 BT5、Kali Linux 下已集成）。首先启动两个重要的服务分别是 MySQL 和 Metasploit　RPC，操作菜单位于：

- Applications→Kali Linux→System Services→Mysql→mysql start
- Applications→Kali Linux→System Services→Metasploit→community /pro start

然后在 Terminal 中输入 armitage 命令，弹出登录界面，显示的主机名、端口、用户及密码都保持默认值，接着单击 Connect 按钮，但系统显示没有发现 RPC 服务器，这时需要选择 Metasploit 的 RPC 服务器，选择 Yes 按钮，稍过一会儿，系统便会启动 Msf 的图形化界面。如图 2-145 所示。

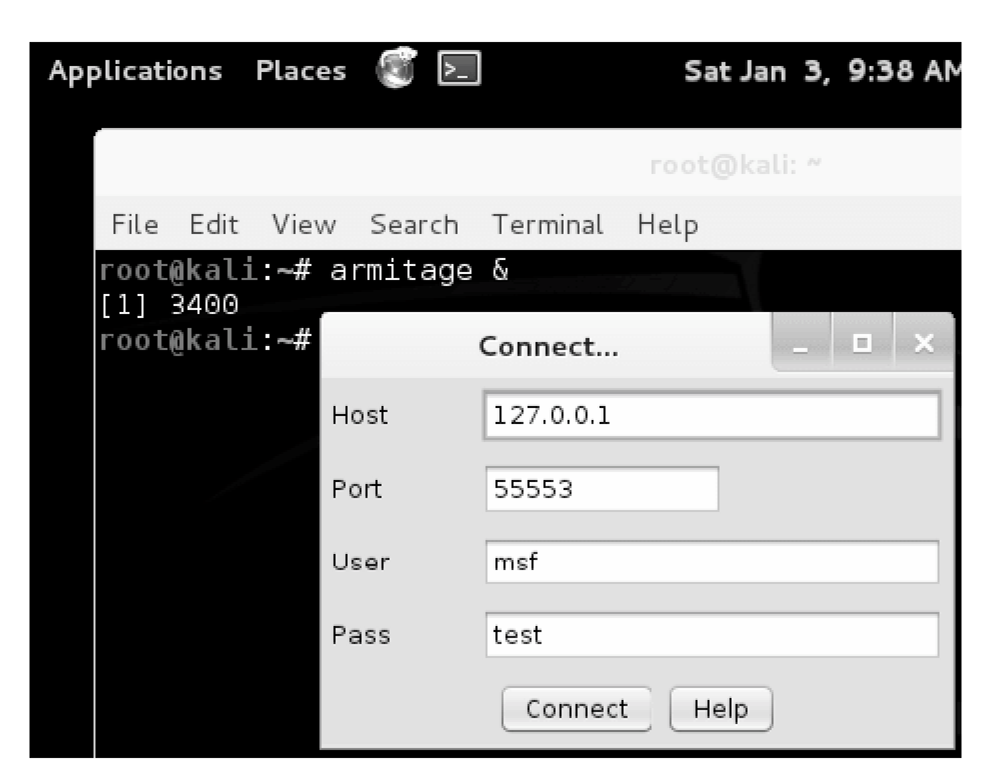


图 2-145　启动 armitage

接着在菜单项中选择 exploits→windows→smb→ms08_067_netapi，将显示配置渗透攻击模块参数的对话框，如图 2-146 所示。

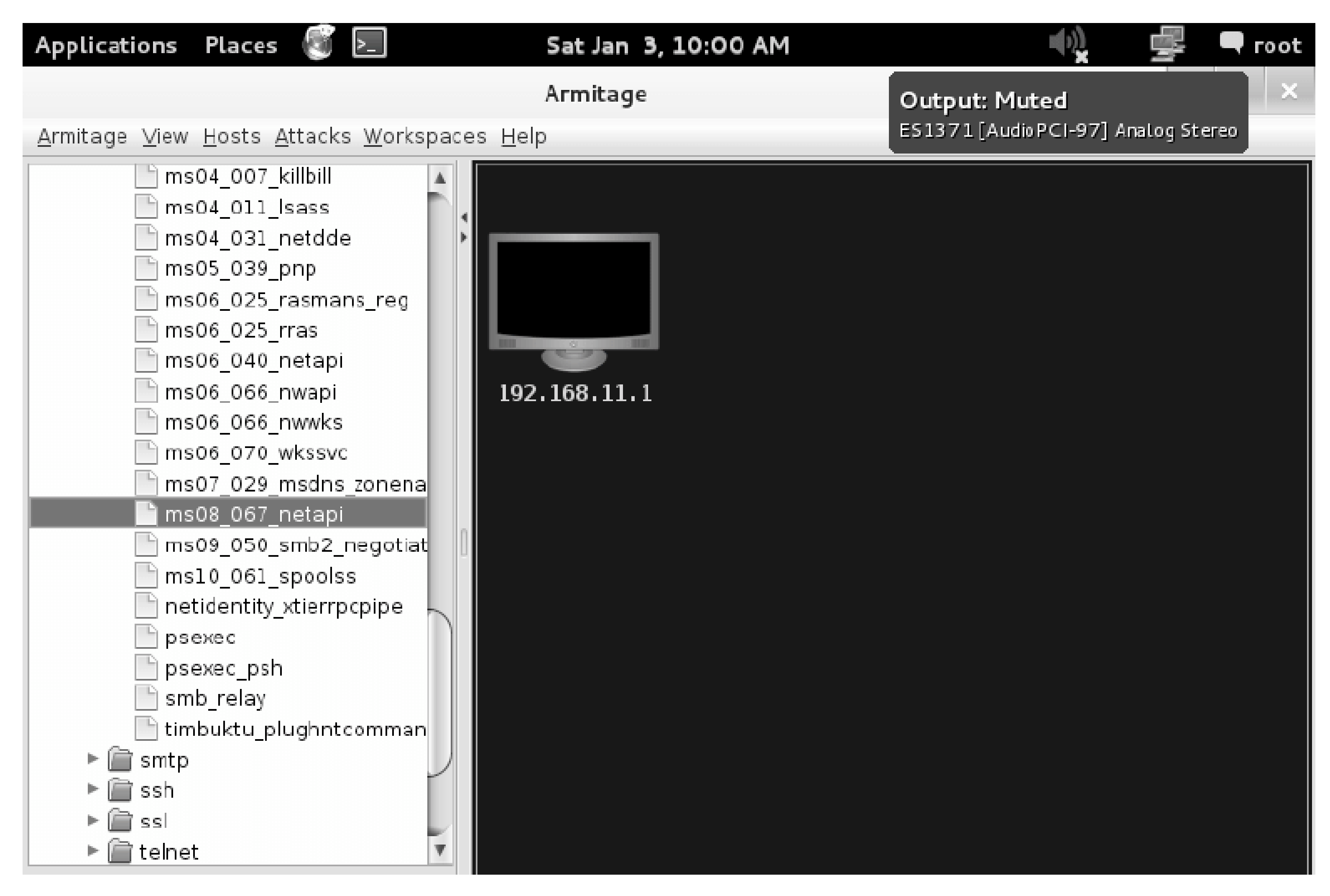


图 2-146　查看渗透参数

此时我们便可对主机进行攻击测试，假设攻击目标 IP 为 192.168.11.121（一台安装英文版的 Windows 2003 Server 系统），首先对目标机进行端口扫描，Armitage 集成了 nmap 工具，单击 Armitage 的“Hosts”菜单，利用“Nmap Scan”这种快速扫描可以自动探测目标主机（或叫做靶机）的操作系统，如图 2-147 所示。在 Armitage 攻击系统时，准确探测操作系统类型、版本比较重要。

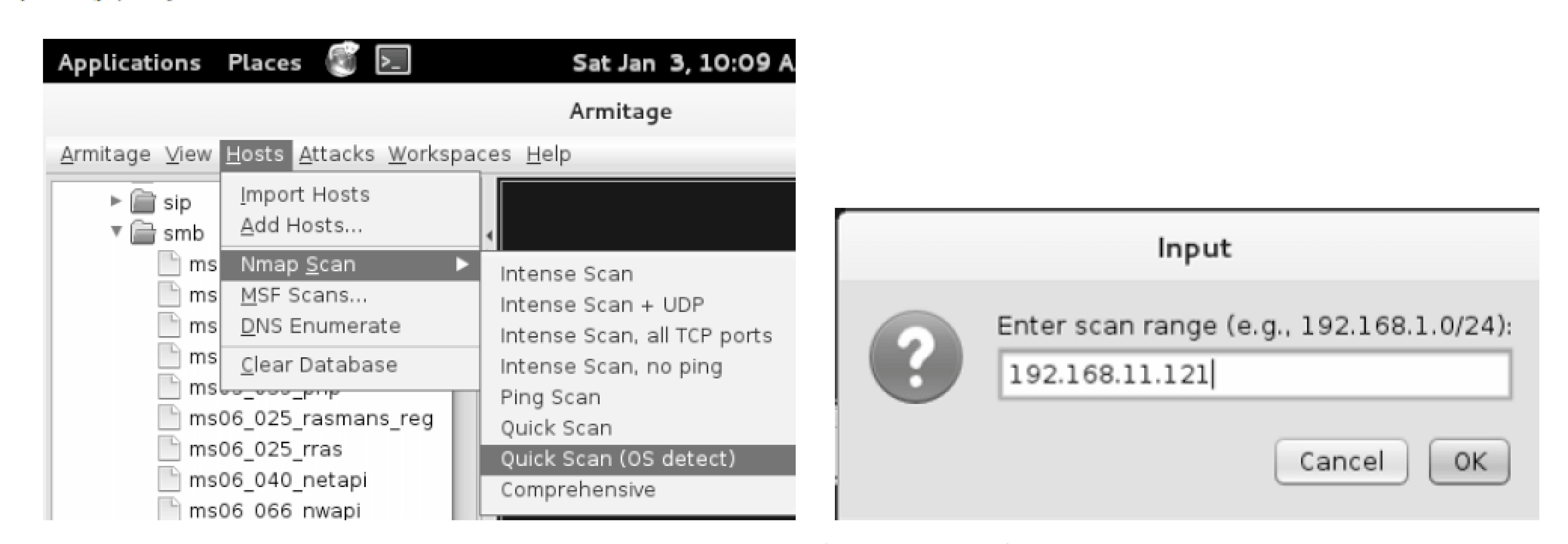


图 2-147　选择扫描方式和对象

扫描完结束后，系统弹出扫描完成对话框，并提示选择“Find Attacks”，之后系统开始分析，出现“Attack Analysis Complete”提示后查找可以利用的 exploit。最后会显示 nmap 的扫描结果，如图 2-148 所示。

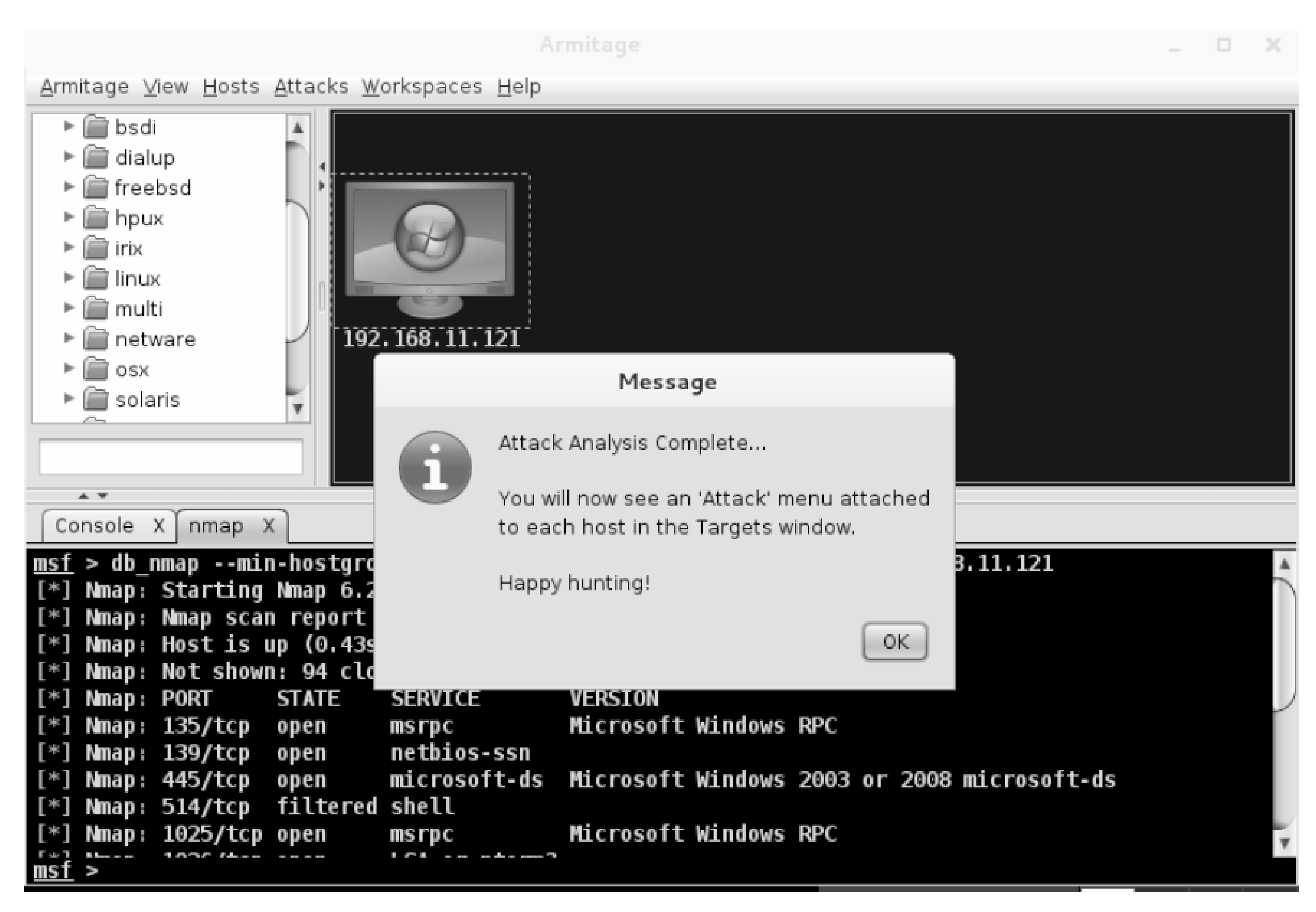


图 2-148　分析目标主机

此时我们选择目标系统的 ms08_067 漏洞，进行渗透，如图 2-149 所示。

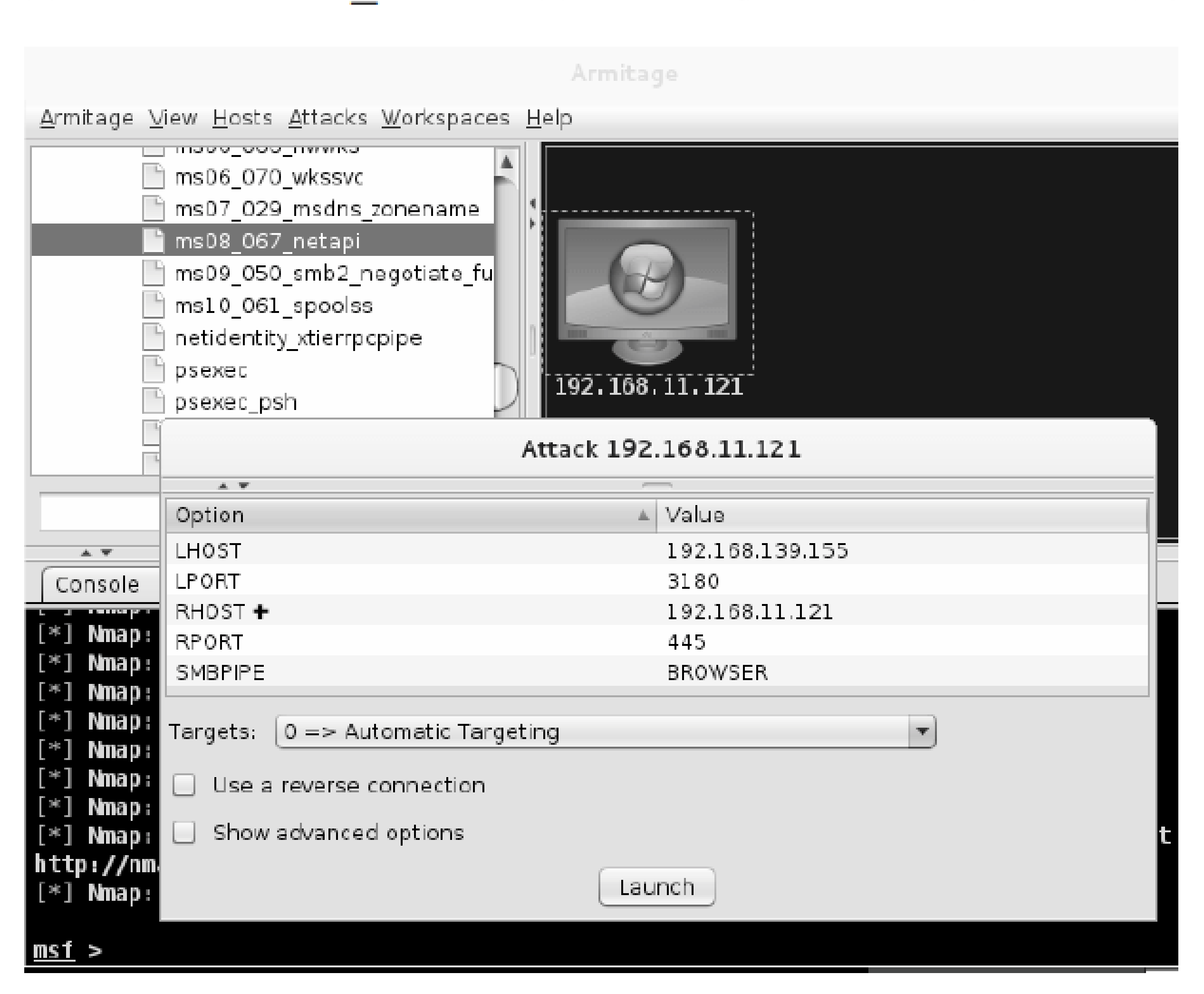


图 2-149　开始渗透目标主机

单击 Exploit 等待数秒后，提示溢出攻击成功，目标主机图标变成了带闪电的红色，在目标主机图标上单击右键选择 Command Shell，系统反弹一个 Shell 表示成功，如图 2-150 所示。

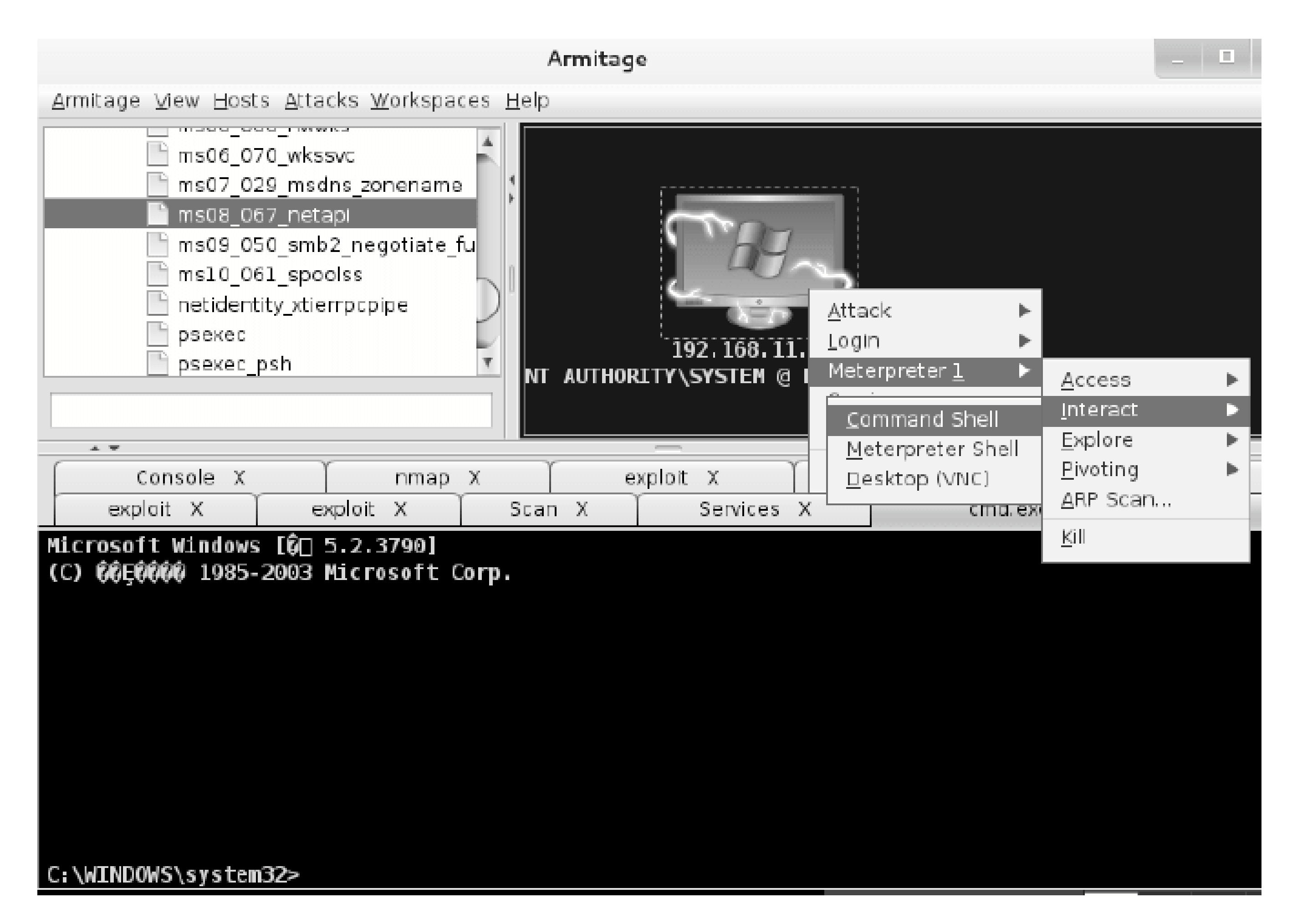


图 2-150　渗透成功

Kali Linux 系统如果出现屏幕保护，需要输入 root 的密码，默认为 toor。

下面再看看控制台下如何实现。

这里实验是通过 OSSIM+Metasploit 联合发掘 Windows XP 系统的 Ms08-067 漏洞，过程如下，以 BT5 系统为例（其他版本同样参照其执行），启动 MSF 终端。

```
#msfconsole
```

1. 升级系统

```
#msfupdate
```

升级完成后，所下载的文件存放在/opt/framework/msf3/目录下。有时候在升级时会遇到错误提示，例如：

```
svn: GET of '/svn/!svn/ver/1609/framework3/trunk/lib/anemone/page.rb':could not connect to server (https://www.metasploit.com)
```

这时，重新执行“msfupdate”即可，升级过程中切勿强行终止升级。

2. Armitage

输入以下命令，系统会打开一个图形化界面的 Metasploit。

```
#armitage
```

3. 添加数据库服务器主机 IP 或网段地址

添加数据库服务器主机 IP 或网段地址，输入:192.168.11.0/24。msfconsole（控制台终端）是 Metasploit 渗透测试框架中的用户界面，在终端输入命令“msfconsole”便可进入控制终端。首先利用 nmap 扫描目标主机，并发现 MS08-067 漏洞，如图 2-151 所示。

图 2-151　控制台下扫描

查看详细信息，执行“info windows/smb/ms08_067_netapi”命令，接下来执行“use windows/smb/ms08_067_netapi”命令，进入到渗透攻击模块之中，再执行“show payloads”命令查看该模块可以使用的攻击载荷。在攻击载荷中选择 reverse_tcp 模块，执行“set payload windows/meterpreter/reverse_tcp”命令将其配置到渗透模块之中。

该载荷的作用是在渗透攻击成功后，执行“reverse_tcp”模块中的 Shellcode，利用这个 Shellcode，创建一个反向链接的会话。当选配置好载荷参数后，执行“show options”命令来查看需要配置的目标参数。在 ms08_067_netapi 模块中，需要设置 RHOST 参数为目标靶机 IP 地址，但 RPORT 参数、LPORT 参数以及 target 参数都可以使用默认值。渗透成功的界面如图 2-152 所示。

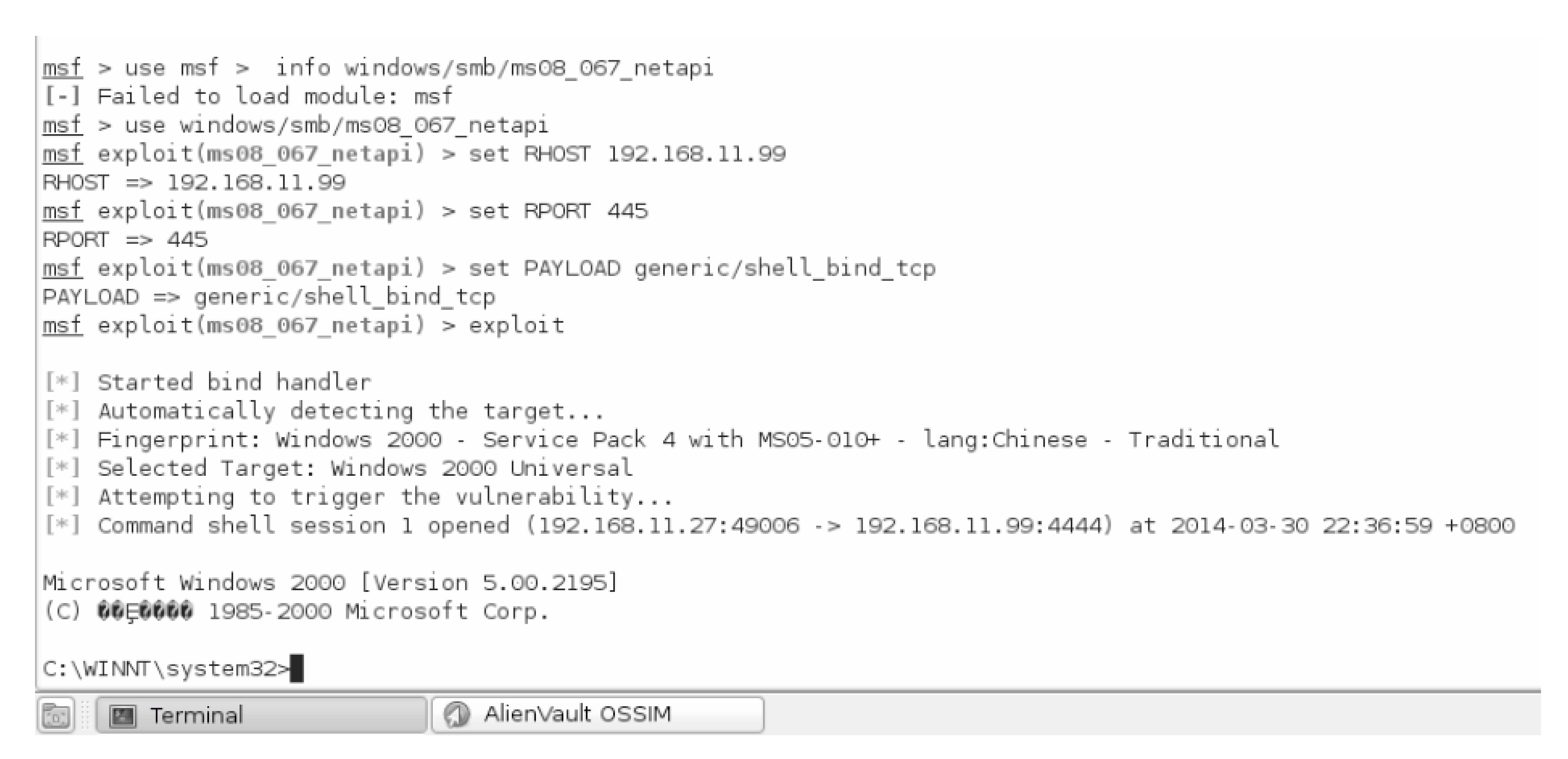


图 2-152　渗透成功

如果攻击者采用这类渗透攻击，那么在 OSSIM 监视下将立刻报警，这时我们查看 SIEM 事件，就能发现报警，如图 2-153 所示。并能查看到渗透攻击的时间线，如图 2-154 所示。

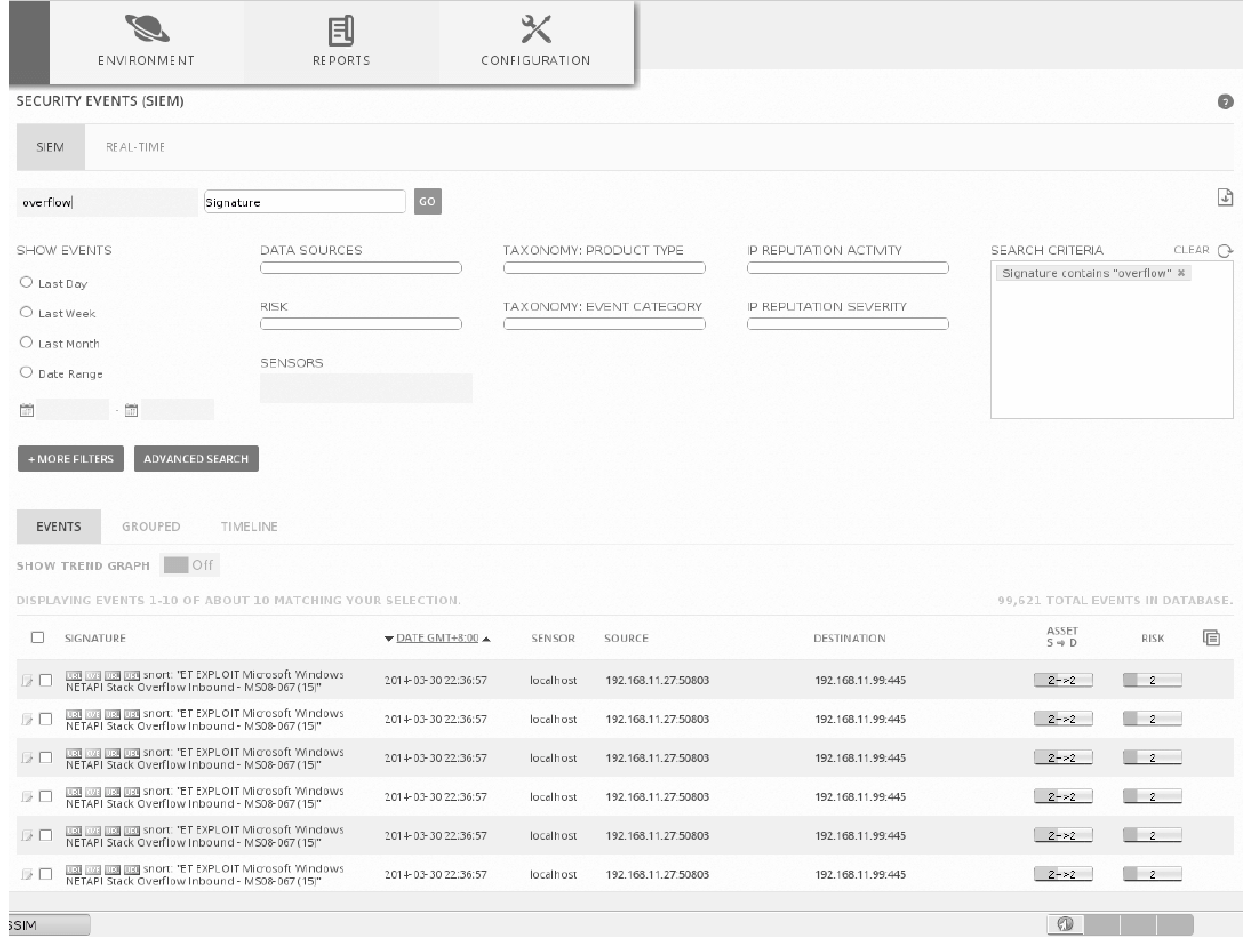


图 2-153　查看 SIEM 报警

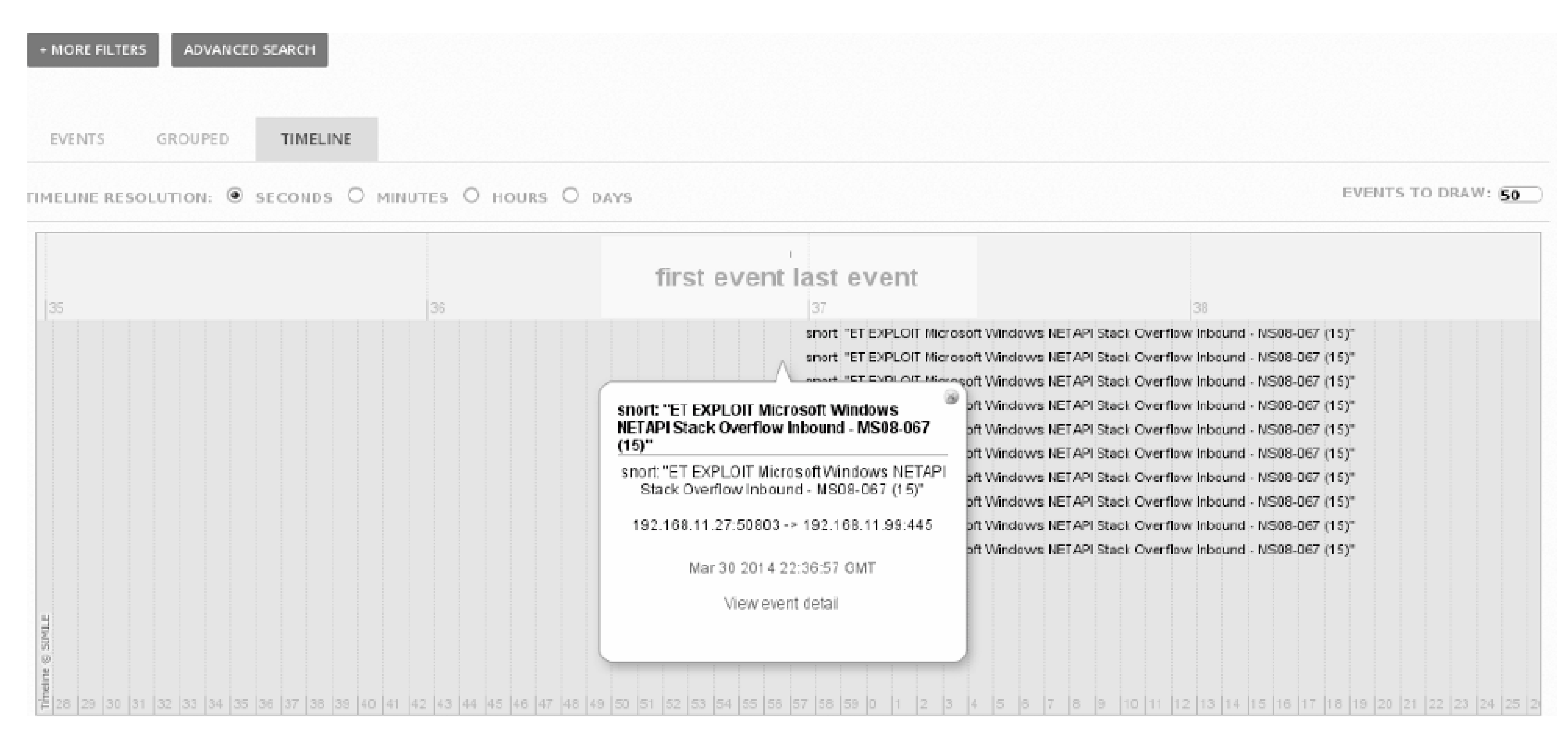
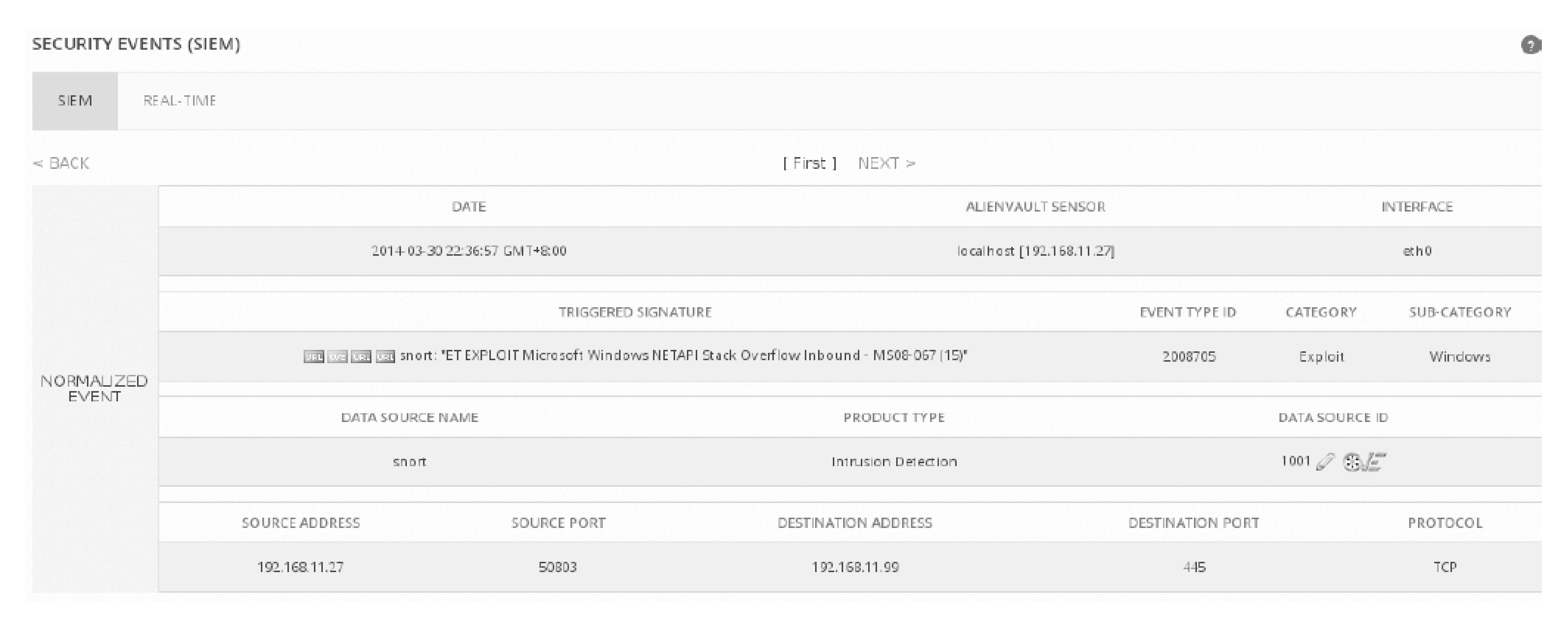


图 2-154　查看渗透攻击时间线

查看某条事件的详细信息，如图 2-155 所示。



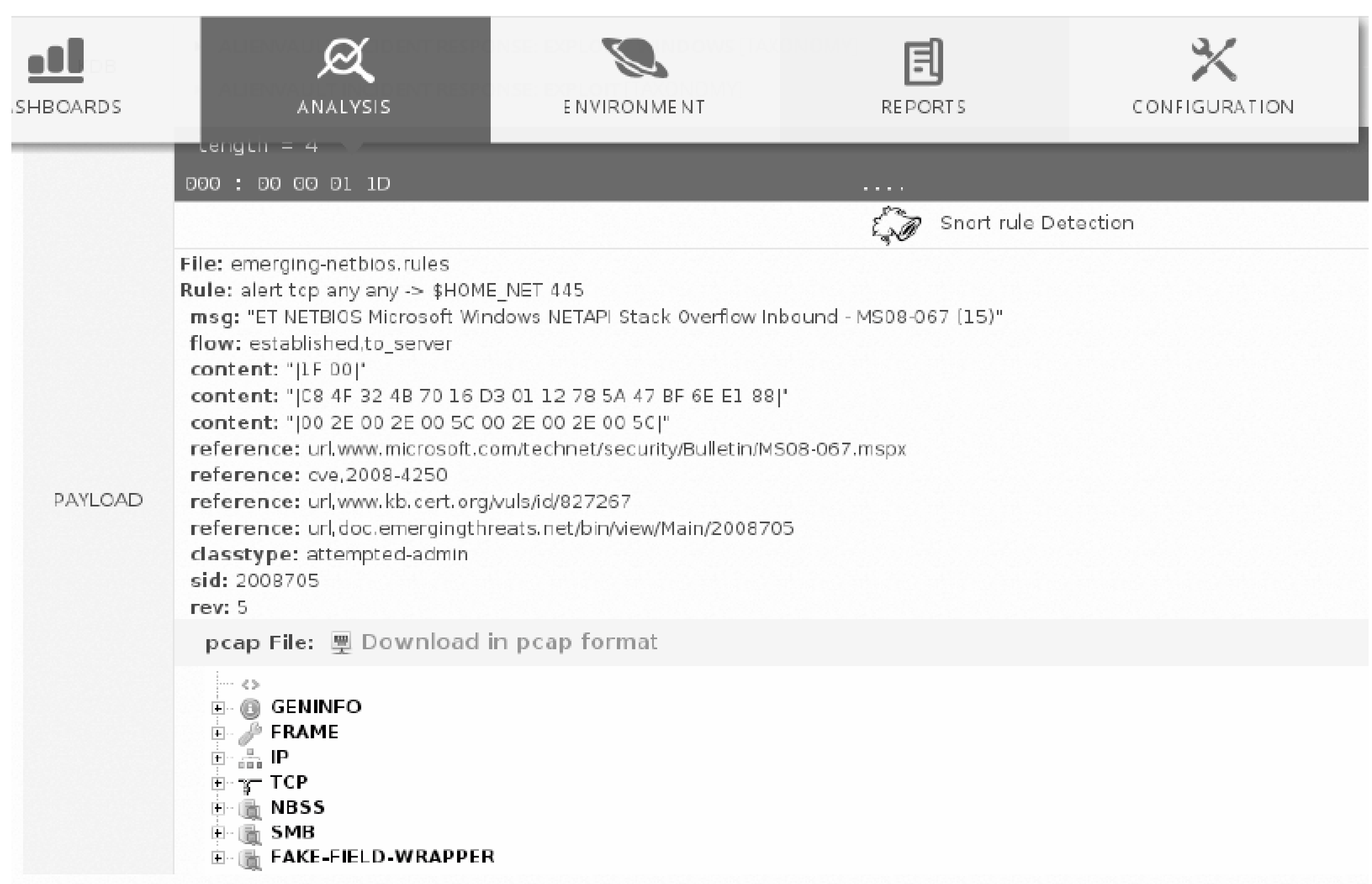


图 2-155　查看某条渗透攻击事件

这里 Snort 系统可以保存恶意流量对应的有效载荷，通过单击“Download in pcap format”按钮下载成 pcap 包，以便再利用其他工具进行分析，例如 CapAnalysis。为了节省篇幅大家可以访问作者的博文：http://chenguang.blog.51cto.com/350944/1325742/，有关漏洞分析检测的内容详见第 9 章。

2.18 小结

本章从 OSSIM 安装策略讲起，逐步讲解 OSSIM 安装前的各项准备工作，对于 OSSIM USM 和 Sensor 之前的区别以及商业版和开源版之间的区别进行了对比，在物理服务器和虚拟机的安装方法也进行了介绍，详细介绍了 OSSIM 混合模式安装，以及分布式安装的方法，对安装之后必须要做的几件事也做了详细阐述，最后详细分析了 SIEM 系统控制台的使用技巧，这对于分析 OSSIM 事件和日志有着非常重要的帮助作用。

看完这些内容，了解了高大上的图表之后，OSSIM 系统本身是否能解决企业面临的安全问题呢？如果仅生成一堆的统计图表的肯定不能算真正意义上 SIEM，要用好 OSSIM 系统还需要结合企业自身的安全团队使用，这才是一个整合资源与流程的自动化处理中心，部署 OSSIM 前先有一个安全团队、一定的资产信息、有一套的信息安全管理流程和安全事件处理流程，部署 OSSIM 系统不是像防火墙、防毒软件那样，买来即用，更多的工作是需要企业安全团队的软实力参与。

第二篇

提 高 篇

Open Source Security
Information Management
Big Data
Cloud Computing
Internet of Things

第 3 章

◄ OSSIM数据库概述 ►

从本章节可以学习到：

- 访问 OSSIM 数据库（本地、远程）
- OSSIM 数据库结构
- OSSIM 中的 MySQL 常见操作
- OSSIM 系统迁移
- OSSIM 常见数据库问题解答

事件关联是整个 OSSIM 关联分析的核心，OSSIM 的事件关联需要海量处理能力，主要体现在需要及时存储从设备采集到的日志，并能关联匹配和输出，进而通过 Web UI 展示。从实时性上看，关联分析的整个处理过程不能间断，这对系统的实时性要求较高，另外 OSSIM 系统是基于规则的，OSSIM 内部具有多套高速规则分析引擎，以实现模式匹配和对关联分析结果调用。所以系统的关联引擎是一个典型数据处理系统，必须依靠强大的数据库做支撑。在开源 OSSIM 系统中，就采用了基于 MySQL 5.6 的数据库，其商业版采用 MonogDB。

普通日志存入数据库较容易，但如果是关联引擎，将告警存入数据库的过程要复杂，到底它的压力在哪儿？例如一个关联规则需要在 1 秒钟内，通过 SQL 语句获取 10 条数据，那么关联引擎就需要在 1 秒钟内进行 10 次磁盘存取，这个要求就比普通日志存入数据库高，而 OSSIM 数据库中的表、字段、索引都为了这种事务处理做了特殊设置，具有一次写多次读的特性。对于复杂模式的匹配非常有用。例如，筛选出 1 分钟内 SSH 登录服务器，失败次数超过 5 次的源 IP 地址，关联分析引擎将定时进行 SQL 访问，找到某个符合要求的事件记录。另外，本章介绍的数据库知识相对容易、无须专业，下面开始介绍 OSSIM 数据库结构、监控及备份方法。

3.1 OSSIM 数据库组成

3.1.1 MySQL

OSSIM 融合了传统和现代数据库技术，即 SQL 和 NoSQL，第 1 章还介绍过 OSSIM 使用 Redis 作为消息队列服务器，Redis 属于 NoSQL 数据库，在商业版 OSSIM 中还使用了

MonogoDB。下面我们先看看 OSSIM 下主要数据库 MySQL 结构。

OSSIM 4.8 在安装最后阶段系统会产生 information_schema、ISO27001An、PCI、alienvault、alienvault_api、alienvault_asec、alienvault_siem、avcenter、categorization、datawarehouse、myadmin、mysql、ocsweb 和 performance_schema 在内的 14 个数据库。

- Datawarehouse: 存储 OTX（Open Threat Exchang），记录公开威胁交换的相关数据。
- Ocsweb: 存储 Ocs 信息，更多 Ocs 内容参见我的博客《Ocs Inventory NG 使用》一文。
- Alienvault: 存储 SIEM 信息。
- Alienvault_api: 存储 alienvault 应用程序接口信息。
- Alienvault_siem: 存储事件信息。
- avcenter: Alienvault 中心数据库，存放着管理和开发组件和 OSSIM UI 信息。
- ocsweb: 在 OSSIM 系统中使用了开源 IT 资产管理系统（Open Computer System Inventory）这个数据库中存放着服务器资产数据。

OSSIM 2.x 3.x、4.0、4.1、4.2 使用的数据库种类如下所示：

```
    information_schema 、ISO27001An 、PCI 、categorization 、datawarehouse 、
jasperserver 、myadmin、 mysql、 ocsweb、 ossim 、ossim_acl、 osvdb 、snort。
```

访问 OSSIM 数据库分两种情况：一种是本地访问，一种是远程访问。

3.1.2 本地访问

在 OSSIM 控制台下连接 MySQL，除了传统的命令访问，还可以通过 ossim-db 命令访问。

```
#ossim-db
```

当然也可以通过下面传统的命令访问：

```
#mysql -u root -p
```

在输入口令后即可连接，下面是几个常见命令：

```
>SHOW DATABASES;                    查看数据库;
>USE 数据库名;                       更改默认使用的数据库;
>SHOW TABLES;                       查看数据库中的表;
>SHOW TABLES;pager more;            如果表太长，一屏无法显示就需要分屏显示;
```

分屏显示方法为：

```
>pager less;
>show tables;
>DESC 表名;                              查看表结构;
>SHOW COLUMNS FROM <table name>;         列出表的列信息
>SHOW INDEX FROM <table name> ;          列出表索引信息
>SHOW STATUS;                            列出 server 状态信息
>SHOW PROCESSLIST;                       查看当前 MySQL 进程
```

show processlist 命令非常实用，有时候 MySQL 经常达到 50%以上或更多，就需要用这个命令看哪个 SQL 语句占用 CPU 资源较多。如图 3-1 所示。

```
mysql> show processlist;
+------+-----------------+-----------------+-----------------+---------+------+------------------------+------------------+-----------+---------------+
| Id   | User            | Host            | db              | Command | Time | State                  | Info             | Rows_sent | Rows_examined |
+------+-----------------+-----------------+-----------------+---------+------+------------------------+------------------+-----------+---------------+
|    1 | event_scheduler | localhost       | NULL            | Daemon  | 5294 | Waiting on empty queue | NULL             |         0 |             0 |
| 1874 | root            | localhost:56215 | alienvault      | Query   |    0 | NULL                   | show processlist |         0 |             0 |
| 2241 | root            | localhost:56593 | alienvault      | Sleep   |    0 |                        | NULL             |         0 |             0 |
| 2242 | root            | localhost:56594 | alienvault_siem | Sleep   |    0 |                        | NULL             |         0 |             0 |
| 2243 | root            | localhost:56595 | alienvault_siem | Sleep   | 1924 |                        | NULL             |         0 |             0 |
| 2252 | root            | localhost:56605 | alienvault      | Sleep   | 1915 |                        | NULL             |         1 |             1 |
| 2253 | root            | localhost:56616 | alienvault      | Sleep   |  140 |                        | NULL             |         0 |             0 |
| 2256 | root            | localhost:56619 | alienvault      | Sleep   |   91 |                        | NULL             |         0 |             0 |
| 2257 | root            | localhost:56620 | alienvault      | Sleep   |  113 |                        | NULL             |         0 |             0 |
| 2259 | root            | localhost:56622 | alienvault      | Sleep   |  109 |                        | NULL             |         0 |             0 |
| 2260 | root            | localhost:56623 | alienvault      | Sleep   |  113 |                        | NULL             |         0 |             0 |
| 2262 | root            | localhost:56625 | alienvault      | Sleep   |  108 |                        | NULL             |         0 |             0 |
| 2263 | root            | localhost:56626 | alienvault      | Sleep   |  111 |                        | NULL             |         0 |             0 |
| 2265 | root            | localhost:56630 | alienvault      | Sleep   |  991 |                        | NULL             |         4 |             8 |
| 2266 | root            | localhost:56631 | alienvault      | Sleep   | 1909 |                        | NULL             |         1 |             1 |
+------+-----------------+-----------------+-----------------+---------+------+------------------------+------------------+-----------+---------------+
15 rows in set (0.00 sec)

mysql>
```

图 3-1　查看进程

上图中各列含义如下：

- id：代表标识，Kill 一个语句时用到；
- user：显示当前用户。如果是非 root 用户，该命令就只显示你权限范围内的 SQL 语句；
- host：显示这个语句是从哪个 IP 地址和端口发出。可用来追踪出问题语句的用户；
- db：显示这个进程目前连接的是哪个数据库；
- command：显示当前连接的执行的命令，分为休眠（sleep）、查询（query）、连接（connect）；
- time：该状态持续的时间，单位是秒；
- state：显示使用当前连接的 SQL 语句的状态，state 只是语句执行中的某一个状态，一个 SQL 语句，已查询为例，可能需要经过 copying to tmp table，Sorting result，Sending data 等状态才可以完成；
- info：显示这个 SQL 语句，因为长度有限，所以长的 SQL 语句就显示不全，但仍然是一个判断问题语句的重要依据。

```
#mysqladmin -i10 processlist extended-status      监控 MySQL 的状态
#mysqldump -u root --password=XXXXXXXX --all-database >/backup/backup.sql  备份所有数据库
```

查看 OSSIM 数据库大小，在 MySQL 中由于 information_schema 存放了其他的数据库的信息，我们只要查询其中相应信息就可得知数据库的大小。

（1）指定 information_schema 数据库。

```
>use information_schema
```

（2）查询所有表中数据的大小。

```
>select concat(round(sum(DATA_LENGTH/1024/1024),2),"MB") as data from TABLES;
```

（3）查看指定数据库的大小，比如说数据库 alienvault。

```
>select concat(round(sum(DATA_LENGTH/1024/1024),2),"MB") as data from TABLES
where table_schema="alienvault";
```

（4）查看指定数据库的表的大小，例如数据库 alienvault 中 alarm 表。

```
>select concat(round(sum(DATA_LENGTH/1024/1024),2),"MB") as data from TABLES
where table_schema="alienvault" and table_name="alarm";
```

还需注意，所有数据库文件默认放置在/var/lib/mysql 目录下，其中 alienvault、osvdb 这两个数据库最大，其容量总共 600MB。

3.1.3 检查、分析表

有关 OSSIM 主要数据库及表结构，请读者访问 http://chenguang.blog.51cto.com/350944/1699995，这里重点讲解 OSSIM 系统中几个重要的表：alienvault.alarm 表、alienvault.event 表、alienvault_siem.acid_event 表，它们存储的数据非常多，而且读写频繁，如果 MySQL 进程在写入中被 Kill，或者系统意外关闭都有可能造成表损坏。MySQL 提供了检查表、分析表和优化表的语句。

```
#ossim-db
mysql>USE alienvault                 /*打开 alienvault 库*/
mysql> CHECK TABLE alarm,event;      /*检查表 alarm 和 event 是否存在错误*/
mysql>ANALYZE TABLE  alarm;          /*分析表 alarm，分析表期间不能更新和插入记录，
多个表之间用逗号分隔*/
mysql>USE alienvault_siem            /*打开 alienvault_sime*/
mysql>CHECK TABLE acid_event;        /*检查表 acid_event 是否存在错误*/
mysql>OPTIMIZE TABLE acid_event;     /*优化表 acid_event*/
```

下面的结果是一个发现写错误的实例：

```
+----------------------------+----------+----------+-----------------------------------------------------------------------+
| Table                      | Op       | Msg_type | Msg_text                                                              |
+----------------------------+----------+----------+-----------------------------------------------------------------------+
| alienvault_siem.acid_event | optimize | note     | Table does not support optimize, doing recreate + analyze instead     |
| alienvault_siem.acid_event | optimize | error    | Error writing file './alienvault_siem/#sql-6d9_209f7.frm' (Errcode: 28) |
| alienvault_siem.acid_event | optimize | status   | Operation failed                                                      |
+----------------------------+----------+----------+-----------------------------------------------------------------------+
```

```
mysql>OPTIMIZE TABLE extra.data;     /*优化表 extra.data*/
```

```
+----------------------------+----------+----------+-----------------------------------------------------------------------+
| Table                      | Op       | Msg_type | Msg_text                                                              |
+----------------------------+----------+----------+-----------------------------------------------------------------------+
| alienvault_siem.extra_data | optimize | note     | Table does not support optimize, doing recreate + analyze instead     |
| alienvault_siem.extra_data | optimize | error    | Error writing file './alienvault_siem/#sql-6d9_209f7.frm' (Errcode: 28) |
| alienvault_siem.extra_data | optimize | status   | Operation failed                                                      |
+----------------------------+----------+----------+-----------------------------------------------------------------------+
3 rows in set, 1 warning (0.05 sec)
```

并非所有优化都能见效，优化表有时还会起到反作用，笔者曾遇到一套出现故障的 OSSIM 系统，该系统负载非常高，只要做一点查询任务，系统就非常卡，处于一种假死状态，重启系

统后能缓和一些，而过不了多久系统就重复出现这种故障，之后我在系统中发现 event、acid_event 表非常庞大，大约记录了 150millions 条记录，而记录它的数据库文件/var/lib/mysql/alienvault_siem/acid_event.idb 和/var/lib/mysql/alienvault_siem/extra_date.idb，其容量分别达到了 70GB 和 149GB。更让我意外的是，在计划任务列表中发现系统每天对一些大型表进行优化，要知道在这种规模的数据库中进行表优化，MySQL 会通过 Optimizer 模块，根据该 SQL 所涉及数据表的相关统计信息进行计算分析，然后再得出最合理最优化的数据访问方式，尤其是 OSSIM 经常需要多表联合查询，所以消耗了大量 CPU。因此，在 SIEM 事件量膨胀过快的实例中，可以将“Active Event Windows(dayes)”的值缩短为 5 天。

3.1.4　启用 MySQL 慢查询记录

上线的 OSSIM 系统事件量非常庞大，有时为了过滤一些敏感字段，往往查询时间比较长，为了在 OSSIM 数据库中找出这些查询较慢的 SQL 查询（执行时间较长），我们需要更深入分析 OSSIM。MySQL 为我们提供了 Slow Query Log 记录功能，它能记录执行时间超过了特定时长（默认 2 秒）的查询。分析 Slow Query Log 有助于帮我们找到“问题”查询。操作方法如下：

```
#vi /etc/mysql/my.cnf
```

找到#Logging and Replication#确保下面三条语句没有被注释掉（前面没有#号）。

- log-slow_queries=/var/log/mysql/mysql-slow.log。
- long_query_time=5　/*超过 5 秒的查询将会被记录*/。
- log-queries-not-using-indexes　/*没有使用索引的查询记录到 slow query 日志*/。

总之，在分析 OSSIM 系统时，建议开启慢查询记录，在 Web UI 中查询各种事件观察图表变化，同时通 tail 来观察 mysql-slow.log 变化，有利于让你搞清楚 OSSIM 中“花销最大”的 SQL 查询。而对于正式上线系统，则建议关闭该功能。下面介绍两款常用慢查询工具。

MySQL 官方工具 mysqldumpslow，该工具主要包括了统计不同慢查询的出现次数，消耗时间和扫描行数等，操作方法如下：

```
#mysqldumpslow -s c -t 20 /var/log/mysql/mysql-slow.log
```

- -s：表示排序。
- c：表示计数。
- -t 20：表示显示前 20 条。

另一款好用工具是 mysqlsla，它输出的数据报表有利于分析慢查询的原因，包括执行频率、数据量、查询消耗等。该工具可以到作者博客下载 mysqlsla-2.03.tar.gz，解压到目录，进入目录后，检查包依赖关系。

```
# perl Makefile.PL
Checking if your kit is complete...
Looks good
```

```
Writing Makefile for mysqlsla
```

看到上面提示说明可以继续安装，下面开始编译，安装过程。

```
# make && make install
```

简单使用：

```
# mysqlsla -lt slow /var/log/mysql/mysql-slow.log
```

3.1.5 远程访问

由于安全需要，默认情况下我们只能在控制台上登录后对 MySQL 进行操作，但实际工作中常常需要远程对数据库操作，除了通过 phpAdmin、Webmin、MySQL-Front（www.mysqlfront.de）工具外还可以进行以下操作，以实现远端访问 MySQL 数据库（在 OSSIM 中 root 依然是 MySQL 的默认用户名，password 的值可以在/etc/ossim/ossim_setup.conf 中找到）。

（1）编辑/etc/ossim/ossim_setup.conf 文件。

此文件中有一个参数 db_ip，默认为 127.0.0.1，不建议修改。

（2）修改 root 的权限。

通常在 MySQL 的安装文件中包含 MySQL 系统库，其中 user 表，使用 username 与 host 做双主键，如果这张表中没有 root，localhost 这一行字段，那么该用户无权限登录 localhost。如果这个时候不修改权限，例如客户机（IP:192.168.150.200）在联机数据库时就会遇到如下问题。

```
Access denied for user 'root'@'192.168.150.200' (using password:YES)
```

这是因为当前用户没有访问 MySQL 的权限所导致的，我们可以采用如下办法修改 root 权限：

```
mysql> grant 权限1,权限2,…权限n on 数据库名.表名 to 用户名@用户地址 identified by
'连接口令';
```

当数据库名称.表名称被*.*代替时，表示赋予用户操作服务器上所有数据库、所有表的权限。用户地址可是 localhost，也可是 Ip 地址、机器名字或域名，还可以用“%”表示从任何地址连接。为来自 IP 地址为 192.168.150.200 的 root 用户分配对任何数据库的任何表进行所有操作的权限，语句如下。

```
mysql>grant all privileges on *.* to 'root’@’192.168.150.200' with
grant option;
mysql>flush privileges;
mysql>exit
```

经过上面两个步骤之后，就可以在其他主机上使用客户端工具登录 MySQL 服务器。如遇到拒绝访问提示请参照本书 5.6 节问题 6 方法处理。

3.1.6　MongoDB

在 OSSIM USM 中采用 MongoDB 存储资产等事件，首先是因为 MongoDB 的高性能，只要数据量控制在 MongoDB 服务器物理内存的大小以内，其插入、查询性能要超过传统关系型数据库不少。在 OSSIM 系统中事件分析和日志收集分析是典型海量数据存储，高并发的事件，如将日志存放在 DB，这可能使数据库瘫痪，MongoDB 的以下特点，使得它可以胜任这一工作需求：

- 可以存入海量数据；
- 能承受高并发；
- 可以使用廉价存储；
- 单服务器稳定性可以满足要求。

由于篇幅限制，有关 OSSIM 系统中的 MongoDB 的使用方法，大家可以参见我的博客 http://chenguang.blog.51cto.com/。

3.1.7　SQLite

SQLite 是一个开源的嵌入式关系数据库，它在 2000 年由 D. Richard Hipp 发布，它减少了应用程序管理数据的开销，SQLite 可移植性好、易使用、高效而且可靠。在 OSSIM 中由于需要一个轻巧、高效的数据库，所以采用了 SQLite3，它在 av_forward 和 Openvas 中发挥着作用，具体数据库位置如下：

```
av_forward     /var/ossim/av_forward/avcache.db
/var/lib/openvas/mgr/tasks.db
/var/lib/openvas/scp-data/scap.db
```

由于篇幅限制，有关 OSSIM 系统中的 SQLite3 的使用方法，大家可以参见博客 http://chenguang.blog.51cto.com/。

3.2　OSSIM 数据库分析工具

数据库服务器需要四项基本资源：CPU、内存、硬盘和网络。如果这四项资源中任何一项性能减弱、不稳定或超负载工作，可能导致整个 OSSIM 服务器的性能降低，轻则导致日志分析无法达到实时性，重则 OSSIM 系统宕机。

为了确保 OSSIM 系统核心 MySQL 服务器能够一直处于正常运行的状态，保持稳定的性能，我们需要通过分析 OSSIM 工作负载来进一步调整数据库。在命令行方式下通过“show processlist”命令，可以查看一些信息，但不方便，而 OSSIM 系统本身提供了 mytop 命令行分析工具能解决这个难题。

```
    #mytop -u root -p xxxxxx -d alienvault        \\*查找 MySQL root 密码的方法还记
得吗？
```

mytop 结果显示类似于 top，在其第三行的“Key Efficiency”就反映出缓存命中率。除了这个命令行工具以外，这里再介绍几款分析 OSSIM 负载的工具，通过图形化的查询检测工具——MySQL Enterprise Monitor 查询分析器，该工具能够捕捉服务器所执行的查询，以降序的方式根据响应时间列出任务列表。它会将消耗资源最多的任务置顶，这样能够引起注意。大家还可以搭配 iotop 工具监控硬盘 I/O 状况，用 htop 监控进程状况，用“iostat –d –x –k 1”、“iostat –d 10 6”等命令监控硬盘 I/O 操作等，这些工具都是 OSSIM 系统自带。有关 mytop 工具在第 5 章还会详细讲解。

3.2.1 负载模拟方法

这里引入基准测试工具 sysbench（此工具在 OSSIM 系统中不带，可以通过 apt-get install sysbench 安装）来测试 OSSIM 数据库的性能。

sysbench 基本测试举例如下：

（1）测试 CPU 的性能

```
    #sysbench  --test=cpu --cpu-max-prime=20000 run
```

（2）线程测试

```
    #sysbench  --test=threads --num-threads=64 run
```

（3）内存（memory）测试

```
    #sysbench --test=memory --memory-block-size=8K --memory-total-size=2G
--num-threads=16 run
```

（4）文件的 I/O 测试

```
    #sysbench --test=fileio --file-total-size=10G prepare  （创建10G 的文件大小）
    #sysbench --test=fileio --file-total-size=20G --file-test-mode=rndwr run  （读
和写20G 的文件输出 I/O 值）
    #sysbench --test=fileio --file-total-size=20G --file-test-mode=rndwr cleanup
（删除刚创建的测试文件）
```

sysbench OLTP 基准使用 OLTP 模拟了事务处理的负荷。我们展示一个百万级数据表的例子：

```
    #sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test
--mysql-user=root
    prepare
    sysbench v0.4.8: multi-threaded system evaluation benchmark
    No DB drivers specified, using mysql
    Creating table 'sbtest'...
```

```
Creating 1000000 records in table 'sbtest'...
```

数据准备完毕，接着运行 8 个并发、60s 内只读的基准测试。

```
#sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test
--mysql-user=root --max-time=60 --oltp-read-only=on --max-requests=0
--num-threads=8 run
```

在 MySQL 网站上有多款不错的、适合 OSSIM 系统数据库性能监控的数据库，例如 Mysql Workbench 和 Mysql Enterprise Monitor 等，网址为 http://www.mysql.com/downloads/。

MySQL Workbench 是可视化数据库设计、监控管理，以及备份的工具集，它分为开源和商业化的两个版本，目前最新版本 6.4。下面的实验是在 Windows XP +SP3+.Net 4.0 环境下安装 Workbench，其运行效果如图 3-2 所示。

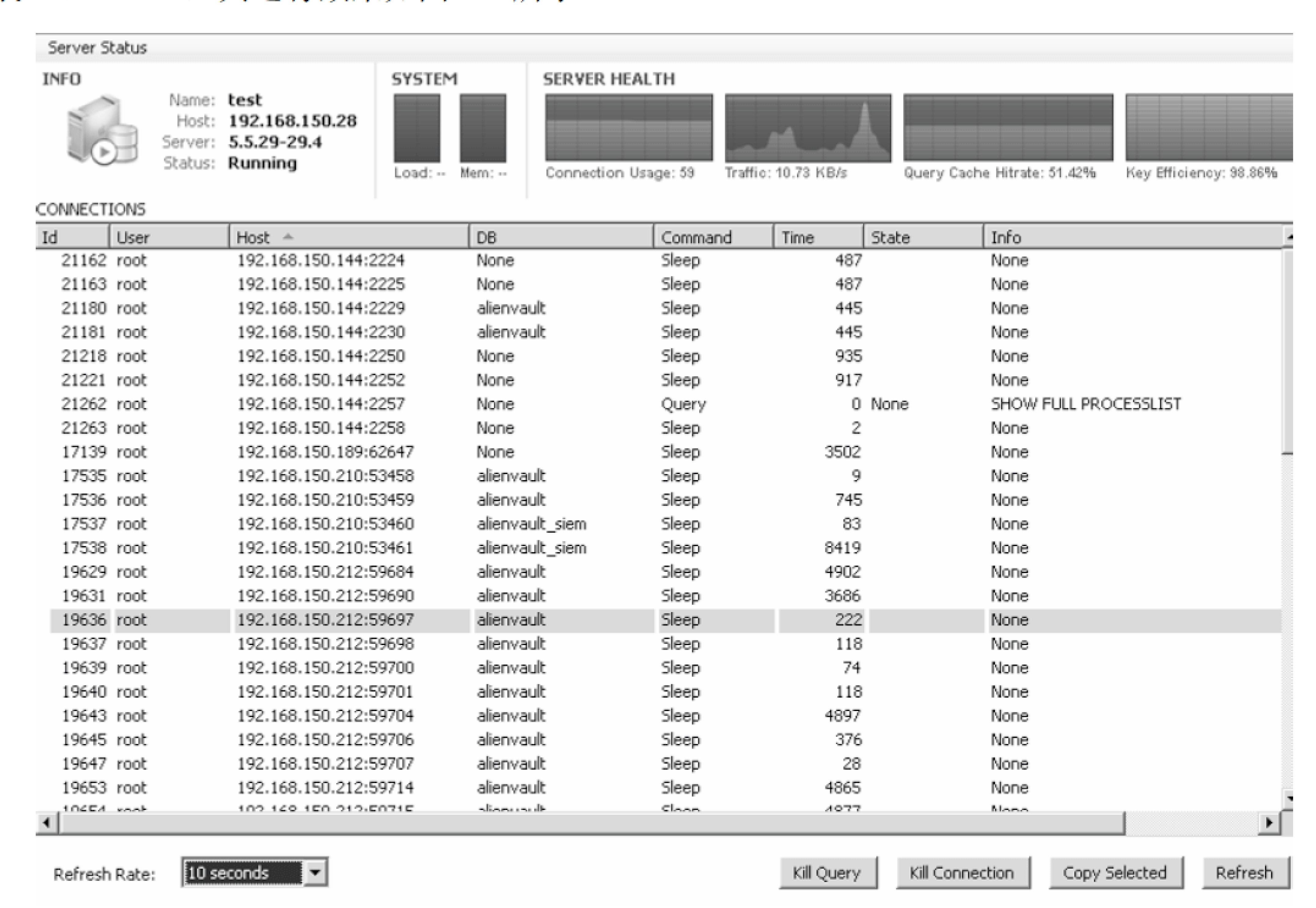


图 3-2　监控 OSSIM 系统各数据库状况

通过这款工具可以方便地将 OSSIM 数据库备份到异地服务器，而无须停止 MySQL 服务，从而保证了系统连续性，其备份过程如图 3-3 所示。

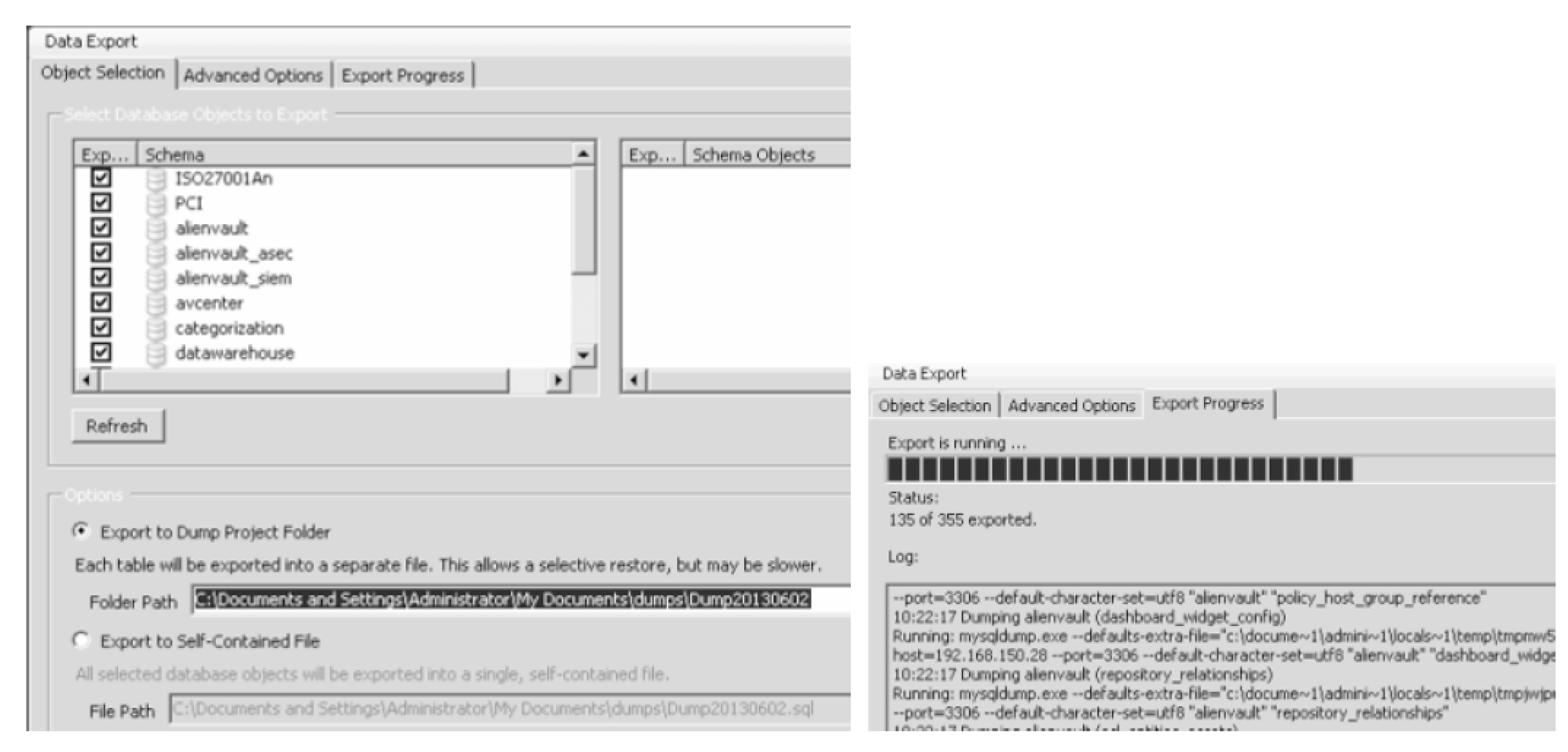


图 3-3　多个数据库导出

另一款可视化数据库分析工具是 MySQL Monitor，它可通过 Web 方式访问，使用方便且能详细地显示数据库工作状况，这里省略它的安装、配置方法。

3.2.2　利用 MySQL Workbench 工具分析数据库

MySQL Workbench 是一款专为 MySQL 设计的 E/R 数据库建模工具。可以用 MySQL Workbench 设计和创建新的数据库图示，建立数据库文档，以及进行复杂的 MySQL 迁移。它分为开源和商业化两个版本。可运行在 Windows、Linux、Mac 平台下，以下操作实例在 Windows XP 下讲解。

目的：利用 Workbench 工具远程分析 OSSIM 数据库的问题。

环境：

- 本机访问虚拟机中的 OSSIM 数据库。
- 本机 IP 为 192.168.0/24 网段。
- 虚拟机：Vmware Workstation 10。
- VM：OSSIM 网卡采用 NAT 模式。
- OSSIM IP：192.168.120.77。
- 网关：192.168.120.1。

建议读者不要使用汉化 MySQL workbench 程序，它在使用时常会出现意想不到的错误。

（1）设置。在 MySQL 中进行如下操作，以便进行远程连接，如图 3-4 所示。

```
MySQL server version for the right syntax to use near 'user' at line 1
mysql> select host,user from user;
+-----------------+------------------+
| host            | user             |
+-----------------+------------------+
| 127.0.0.1       | osvdb            |
| 127.0.0.1       | root             |
| 192.168.120.77  | osvdb            |
| ::1             | root             |
| alienvault      | root             |
| localhost       | debian-sys-maint |
| localhost       | root             |
+-----------------+------------------+
7 rows in set (0.00 sec)

mysql> grant all privileges on *.* to 'root'@'192.168.120.1' identified by 'root' with grant option
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> select host,user from user;
+-----------------+------------------+
| host            | user             |
+-----------------+------------------+
| 127.0.0.1       | osvdb            |
| 127.0.0.1       | root             |
| 192.168.120.1   | root             |
| 192.168.120.77  | osvdb            |
| ::1             | root             |
| alienvault      | root             |
| localhost       | debian-sys-maint |
| localhost       | root             |
+-----------------+------------------+
8 rows in set (0.00 sec)
```

图 3-4　修改权限

（2）为了调试方便，先将 iptables 规则临时关闭。接下来，我们在 Windows 上安装 Workbench，开始备份 OSSIM 数据。首先新建连接，名称为 ossim_db，连接方法采用标准的 TCP/IP 协议，然后输入 OSSIM 主机的 IP 地址“192.168.120.77”，数据库端口 3306，登录用户 root，输入口令，如图 3-5 所示。

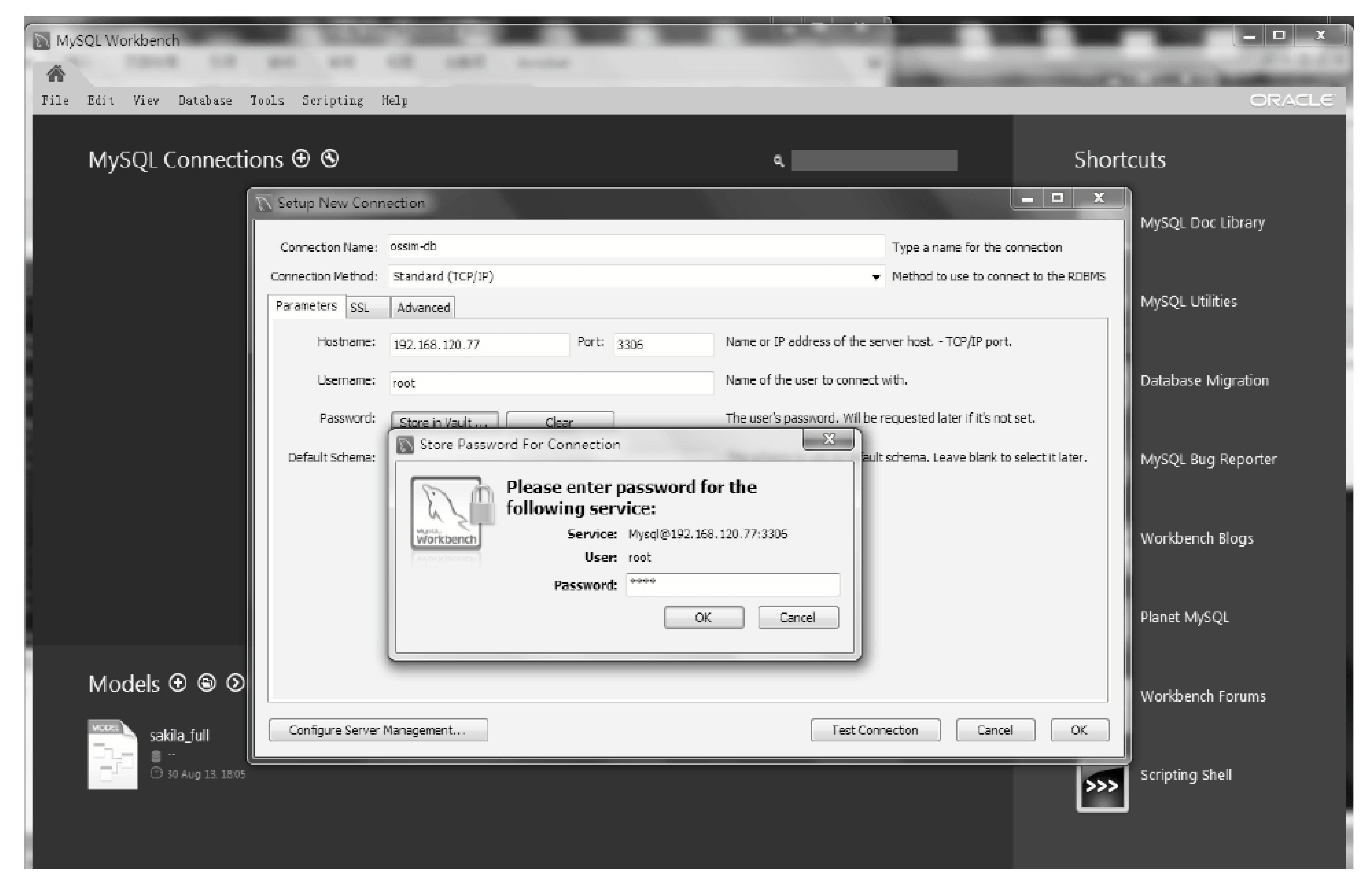


图 3-5　登录界面

（3）系统显示导出数据库进程，如图 3-6 所示，大家要留意导出路径。

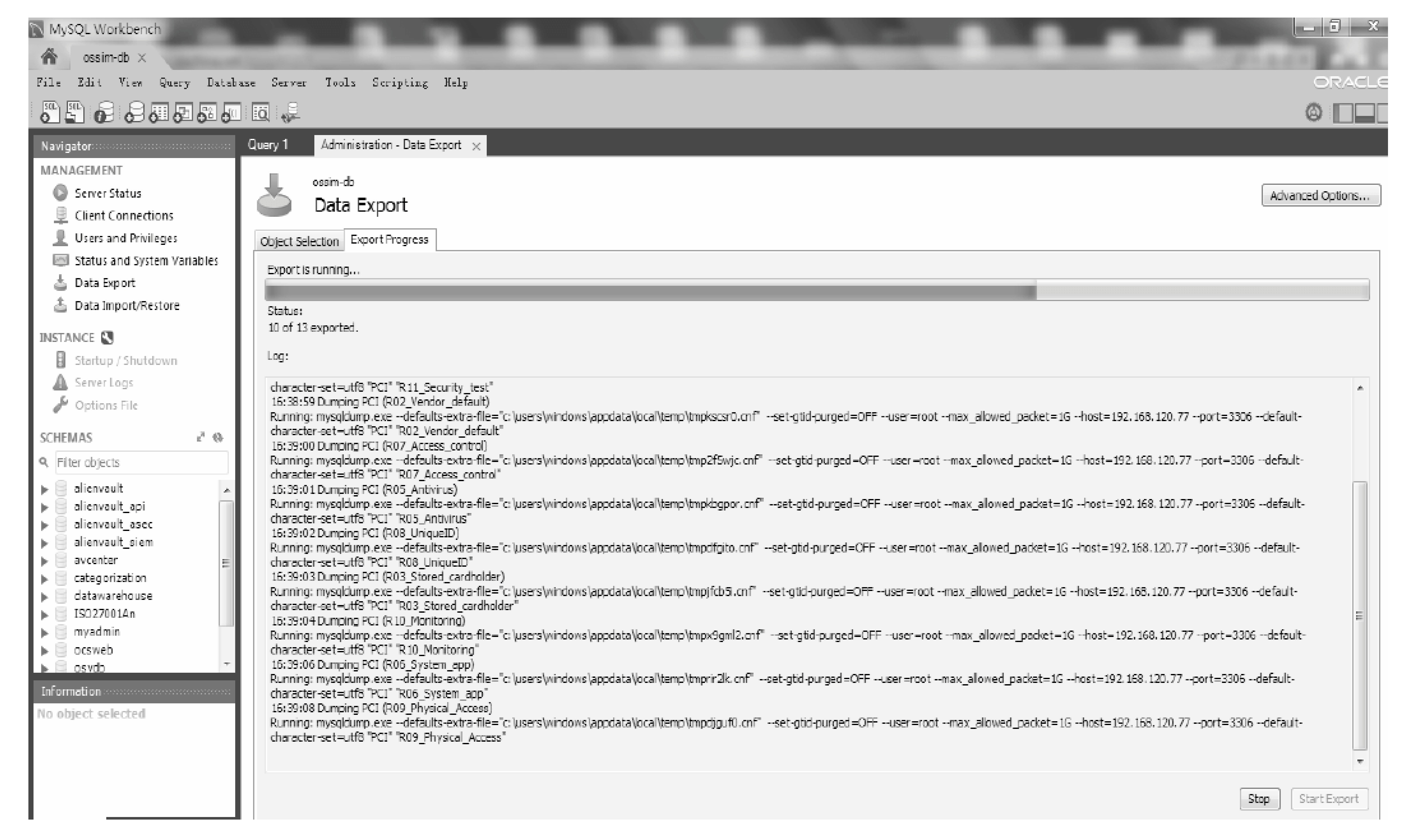


图 3-6　备份数据库

导出成功之后，接着尝试恢复数据库，导入数据库的过程较容易，如图 3-7 所示。

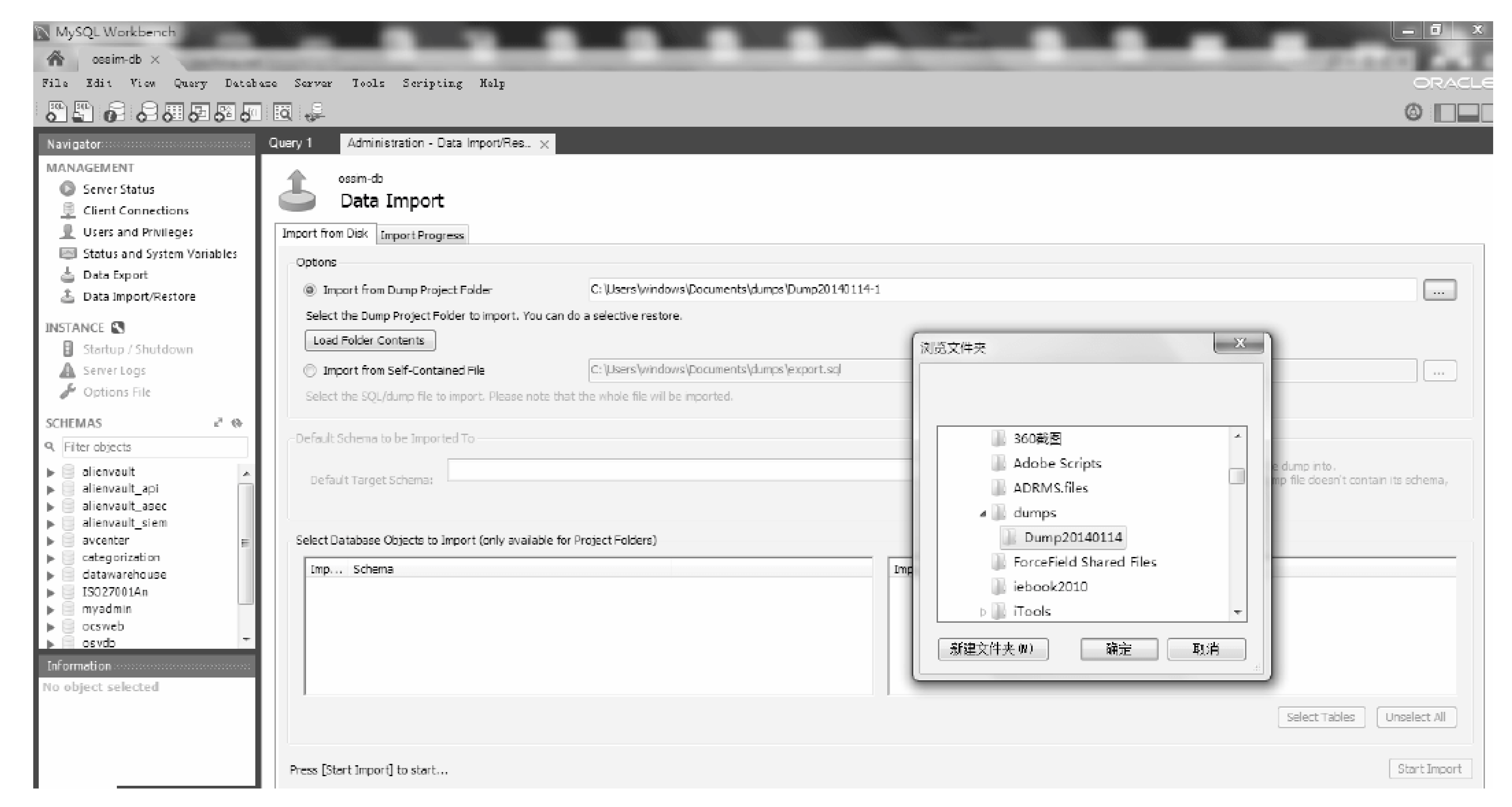


图 3-7　导入数据库

（4）分析 OSSIM 数据库关系（绘制 E-R 图）。

选择 Database 菜单中“Reverse Engineer”按钮，注意先不急于连接数据库。如图 3-8 所示。

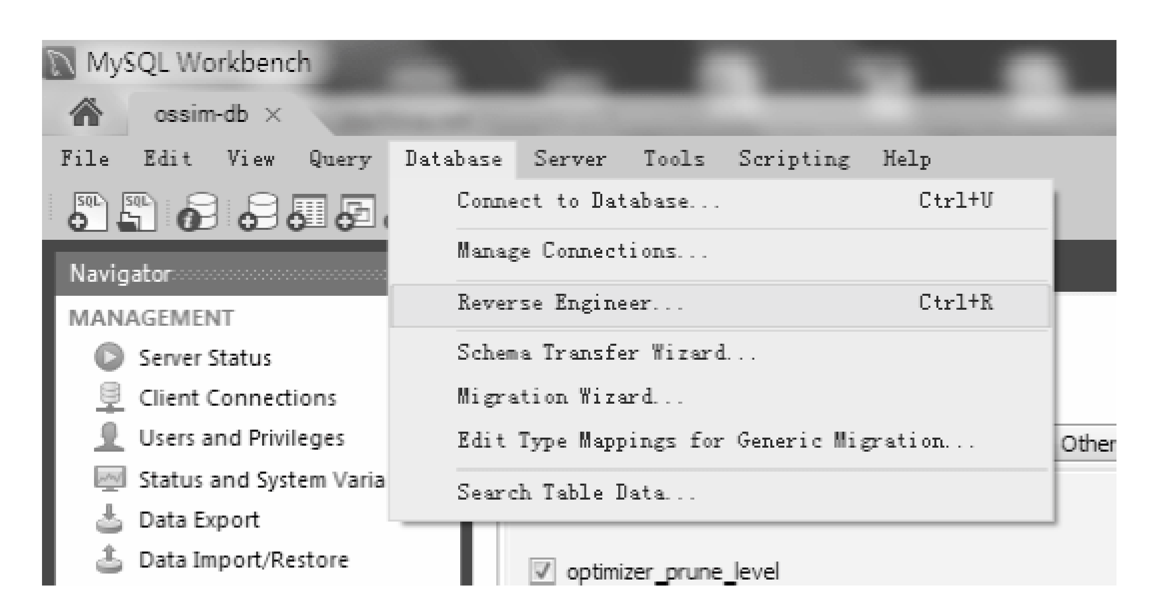


图 3-8　Reverse Engineer

配置参数，输入主机 IP 地址和端口，如图 3-9 所示。

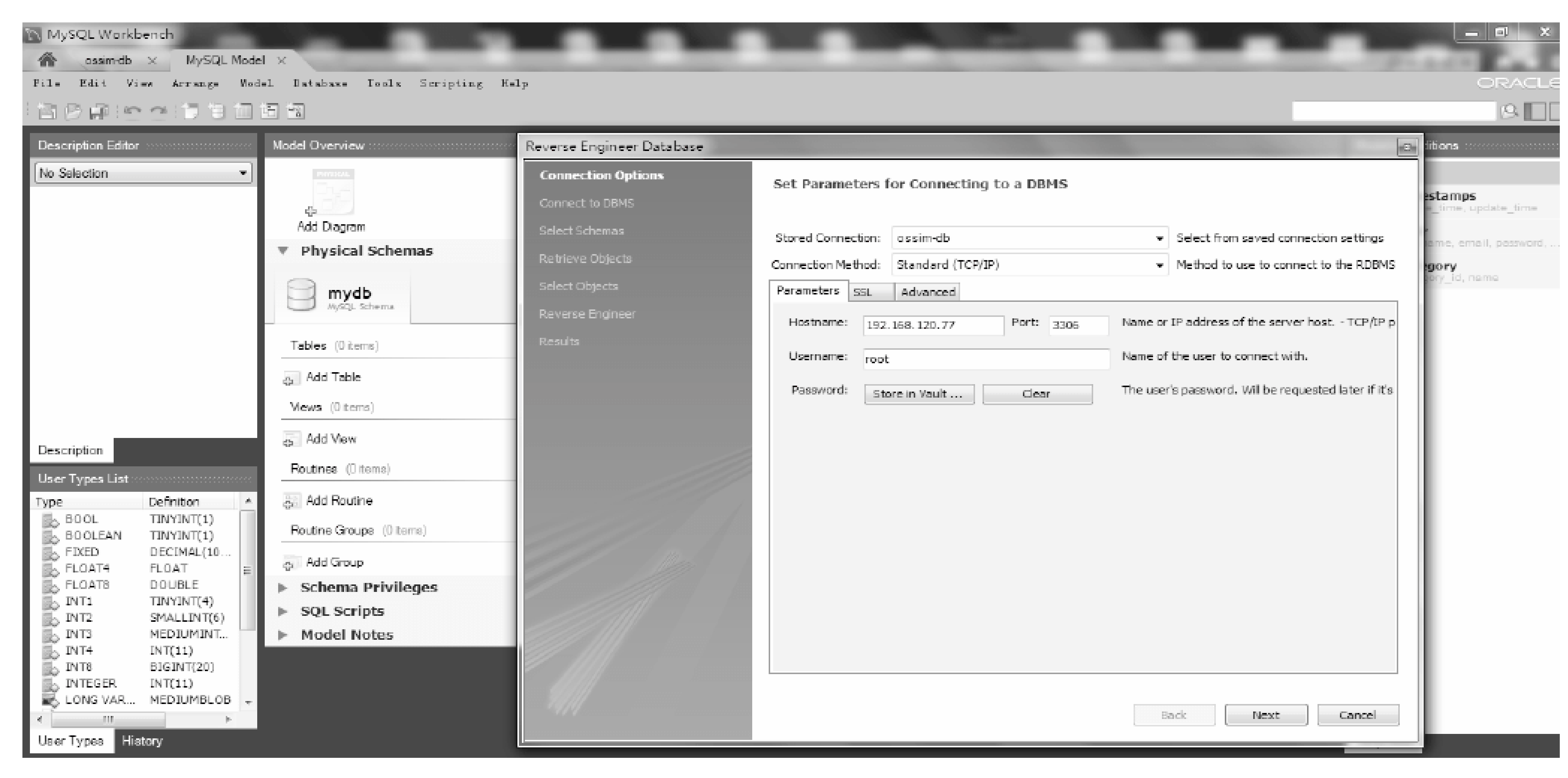


图 3-9　配置主机 IP 和端口

继续选取“Connect to DBMS”和“Retrieve Schema List from Database”，再选择下一步按钮如图 3-10 所示。

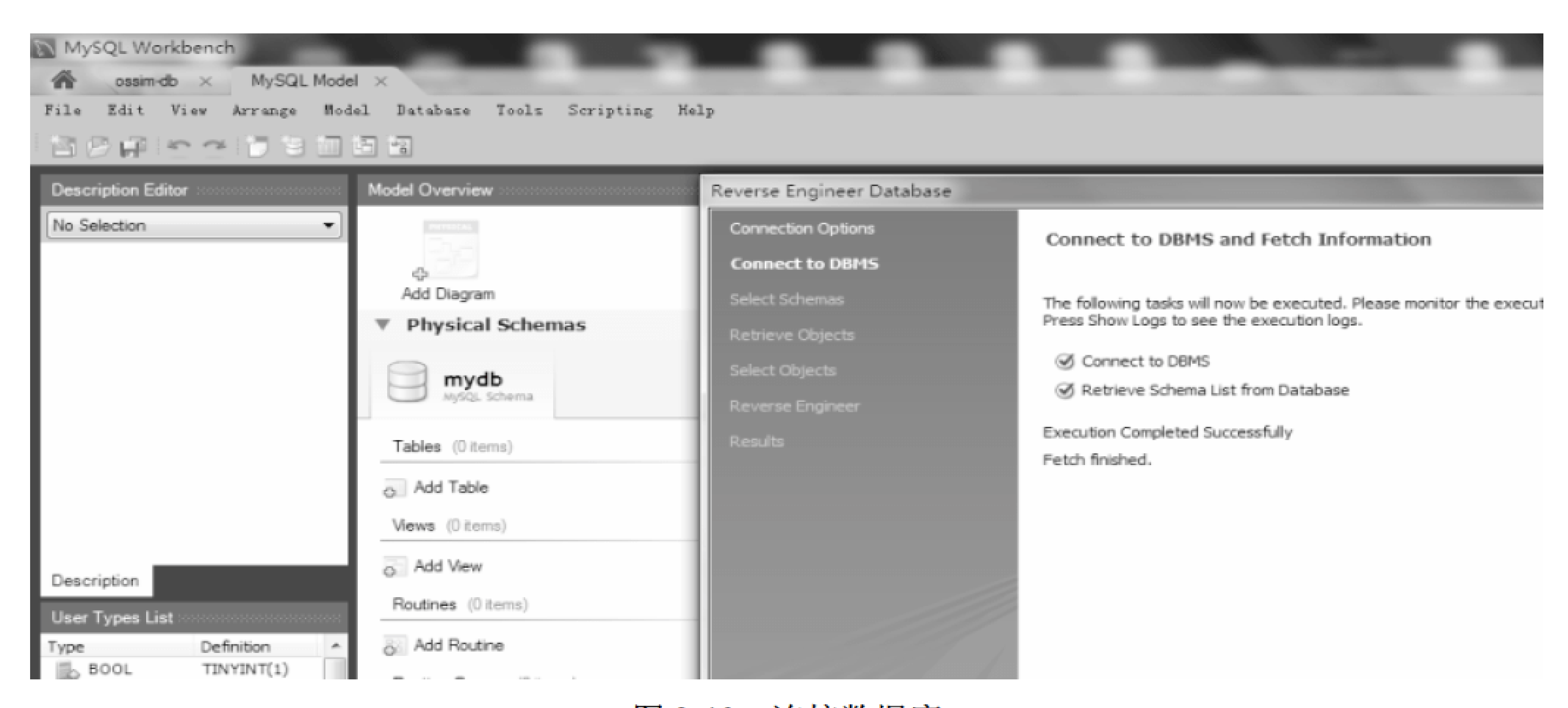


图 3-10　连接数据库

OSSIM 4.8 中存在 12 个数据库，根据需要选择一个或多个，本实例中选择“avcener”，如图 3-11 所示。

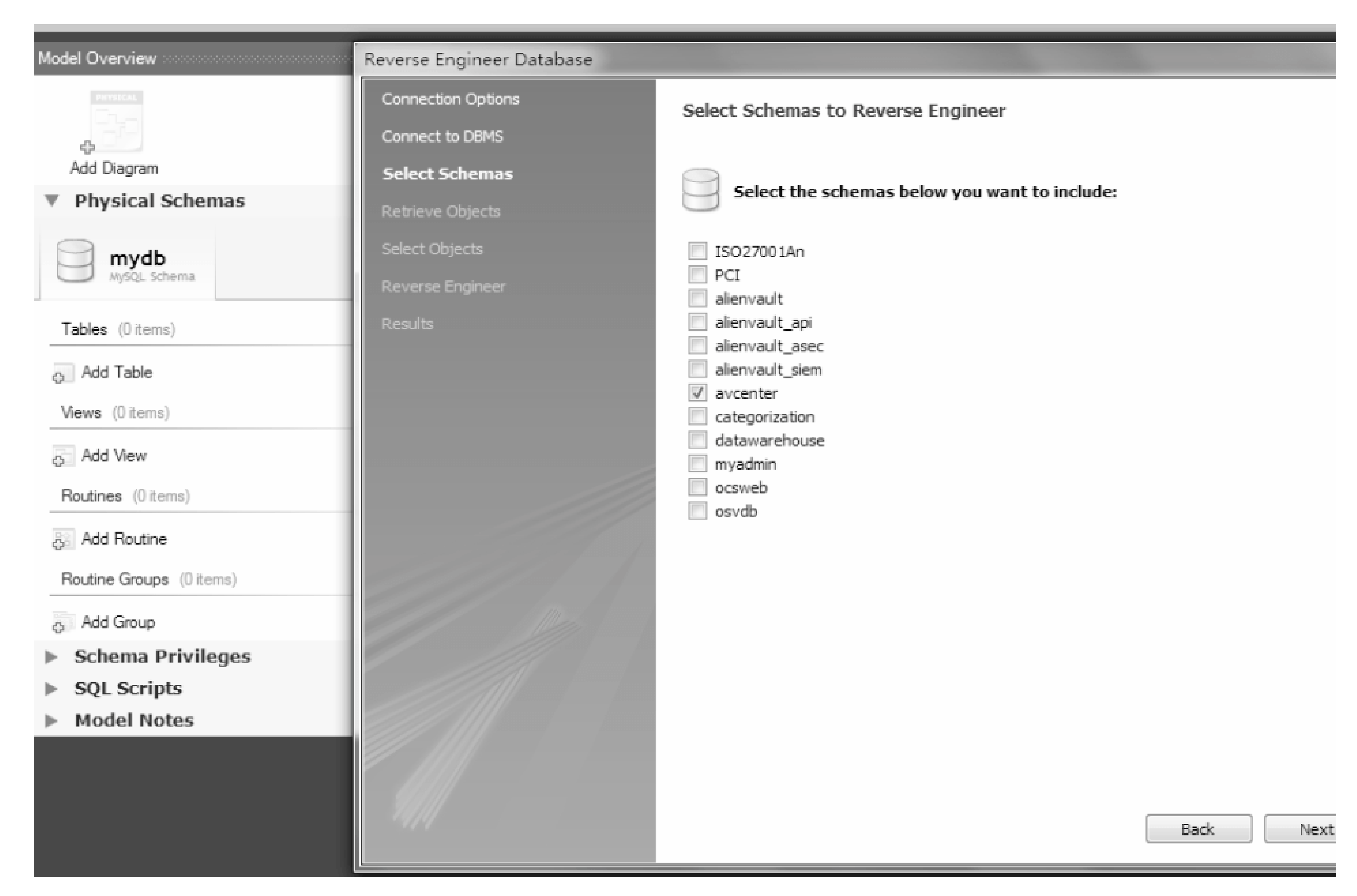


图 3-11　选择数据库

然后选择目标引擎，如图 3-12 所示。

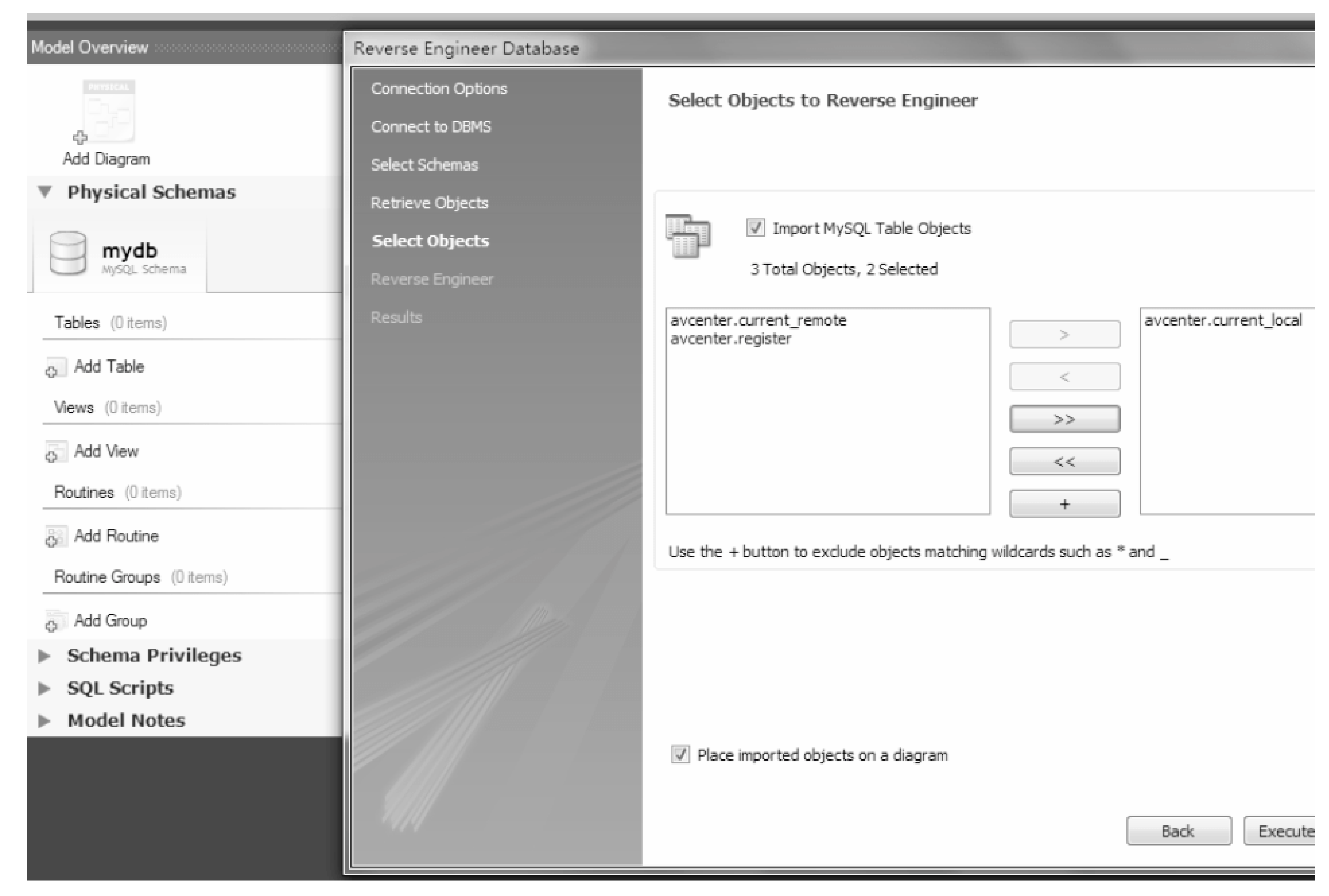


图 3-12　选择目标引擎

最后一步就是设置关联，选择好数据库后按提示操作，如图 3-13 所示。

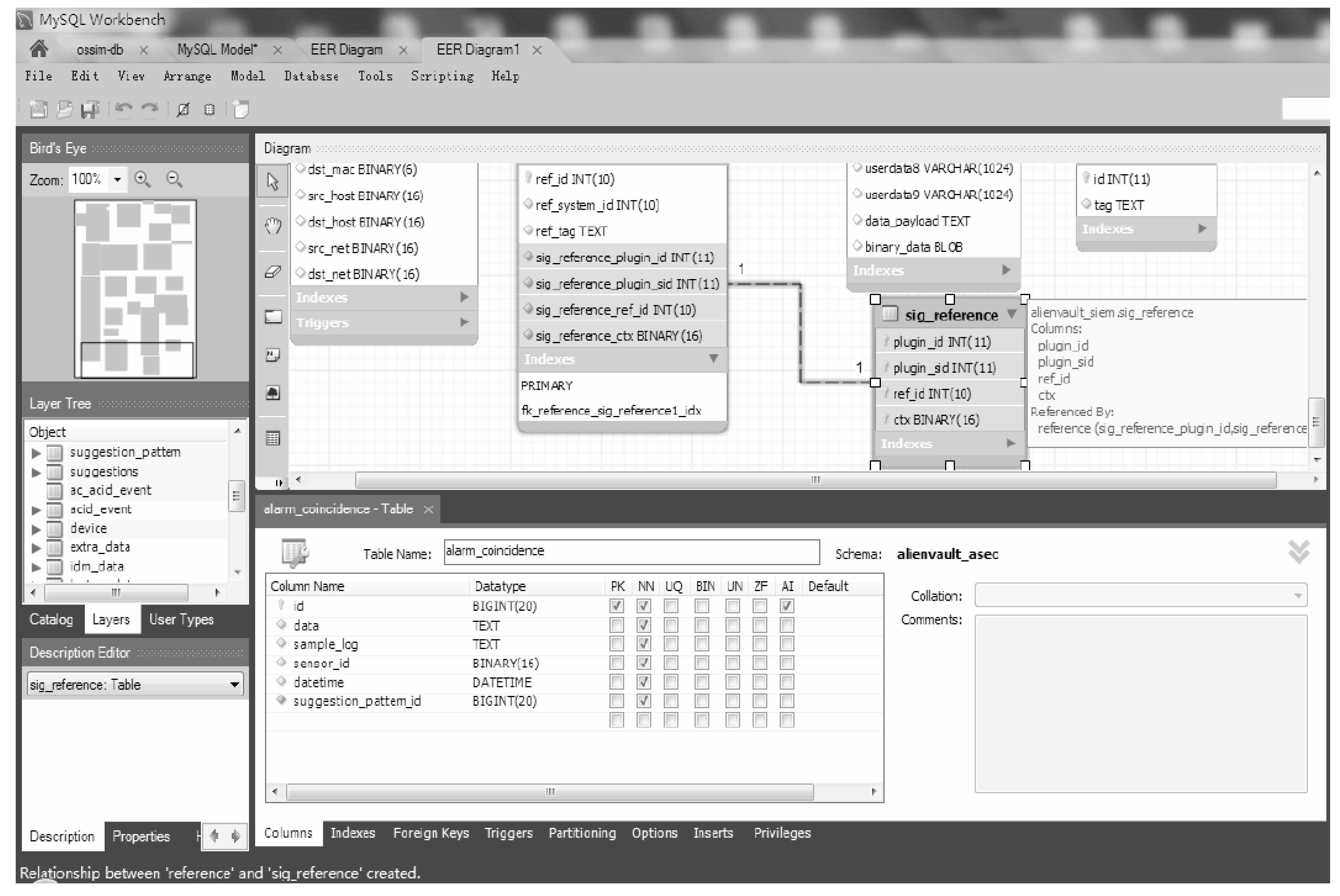


图 3-13　查看数据库关联

分析 OSSIM 数据库设计的 E-R 模型并不难，只要大家熟练掌握 MySQL Workbench 这款工具的使用即可实现。

3.3 查看 OSSIM 数据库表结构

从 OSSIM 数据库的表结构，大家可以了解资源的数据结构，了解 OSSIM 数据库结构对于二次开发的用户、数据库性能调优尤为重要。在作者博客（http://chenguang.blog.51cto.com/350944/1682706/）中列出了 Alienvault OSSIM 4.8 版系统的数据库结构，在更高的 OSSIM 版本中数据库主要结构依然不变。

OSSIM 针对每一个资源都有一张表与其对应，比如 host、alarm……每张表中都有个 id 字段，比如 host 表中有 hostid 列，资源之间的关联通过该 ID 实现。接下来，我们讲解一下对 alienvault 和 alienvault_siem 数据库表的基本操作，如表 3-1 所示。

表 3-1　alienvault、alienvault_siem 主要表查询操作说明

数据库	表	查询操作	查询用途
alienvault	acl_perm	SELECT * FROM `acl_perm` LIMIT 0 , 30	查询 WebUI 菜单权限设置
	alarm_categories	SELECT * FROM `alarm_categories` LIMIT 0 , 30	查询报警分类
	alarm_kingdoms	SELECT * FROM `alarm_kingdoms` LIMIT 0 , 30	查询 Alarm 中 5 种攻击分类
	alarm_taxonomy	SELECT * FROM `alarm_taxonomy` LIMIT 870 , 30	查询 Alarm 分类和子类，用于和 alarm_kingdoms 表关联
	category	SELECT * FROM `category` LIMIT 0 , 30	查询事件分类
	subcategory	SELECT * FROM `subcategory` LIMIT 90 , 30	查询子类
	config	SELECT * FROM `config` LIMIT 0 , 30	查询系统路径、端口、式样及用户配置
	custom_report_types	SELECT * FROM `custom_report_types` LIMIT 0 , 30	查询自定义报表类型及文件位置
	dashboard_custom_type	SELECT * FROM `dashboard_custom_type` LIMIT 0 , 30	查询仪表盘类型参数定义，包括图片位置
	dashboard_widget_config	SELECT * FROM `dashboard_widget_config` ORDER BY `id` ASC LIMIT 0 , 30	查询可拖动窗口配置
	device_types	SELECT * FROM `device_types` LIMIT 0 , 30	查询设备类型记录
	event	SELECT * FROM `event` LIMIT 0 , 30	查询事件记录
	extra_data	SELECT * FROM `extra_data` LIMIT 0 , 30	查询从事件中提取数据，包括 data_payload 等
	Host	SELECT * FROM `host` LIMIT 0 , 30	查询主机记录
	host_ip	SELECT * FROM `host_ip` LIMIT 0 , 30	查询 IP 地址（用十六进制表示）
	host_mac_vendors	SELECT * FROM `host_mac_vendors` LIMIT 8130 , 30	查询网卡厂家 MAC 地址分配
	host_qualification	SELECT * FROM `host_qualification` LIMIT 120 , 30	查询主机记录（包括数据泄露和攻击值）
	net	SELECT * FROM `net` LIMIT 0 , 30	查询网络设置
	plugin	SELECT * FROM `plugin` LIMIT 0 , 30	查询插件名称（包括厂家及描述）
	plugin_reference	SELECT * FROM `plugin_reference` LIMIT 0 , 30	查询插件及引用
	plugin_sid	SELECT * FROM `plugin_sid` LIMIT 0 , 30	查询插件 ID 分配
	port	SELECT * FROM `port` LIMIT 0 , 30	查询服务端口定义
	product_type	SELECT * FROM `product_type` LIMIT 0 , 30	查询产品类型
	protocol	SELECT * FROM `protocol` LIMIT 0 , 30	查询协议及描述

（续表）

数据库	表	查询操作	查询用途
alienvault	repository	SELECT * FROM `repository` LIMIT 0 , 30	查询知识库
	repository_relationships	SELECT * FROM `repository_relationships` LIMIT 0 , 30	查询知识库关系
	risk_indicators	SELECT * FROM `risk_indicators` LIMIT 0 , 30	查询风险指标设定
	rrd_config	SELECT * FROM `rrd_config` LIMIT 0 , 30	查询 RRD 配置
	sensor	SELECT * FROM `sensor` LIMIT 0 , 30	查询 Sensor 记录
	server	SELECT * FROM `server` LIMIT 0 , 30	查询 Server 记录
	server_role	SELECT * FROM `server_role` LIMIT 0 , 30	查询 Server 规则
	tags_alarm	SELECT * FROM `tags_alarm` LIMIT 0 , 30	查询 Alarm 标签
	users	SELECT * FROM `users` LIMIT 0 , 30	查询系统用户
	vuln_nessus_family	SELECT * FROM `vuln_nessus_family` LIMIT 0 , 30	查询 Nessus 族
	vuln_nessus_plugins	SELECT * FROM `vuln_nessus_plugins` LIMIT 0 , 30	查询 Nessus 插件描述（升级后会增加条目）
	vuln_nessus_plugins_feed	SELECT * FROM `vuln_nessus_plugins_feed` LIMIT 0 , 30	查询 Nessus feed
	vuln_nessus_preferences_defaults	SELECT * FROM `vuln_nessus_preferences_defaults` LIMIT 0 , 30	查询 Nessus 默认偏好设置（升级后会增加条目）
	vuln_nessus_settings	SELECT * FROM `vuln_nessus_settings` LIMIT 0 , 30	查询 Nessus 扫描策略
	vuln_nessus_settings_preferences	SELECT * FROM `vuln_nessus_settings_preferences` LIMIT 0 , 30	Nessus 偏好设置（升级后会增加条目）
alienvault_siem	acid_event	SELECT * FROM `acid_event` LIMIT 0 , 30	查询 acid 事件记录
		SELECT COUNT(*) FROM acid_event WHERE plugin_id=4005	统计 acid_event 表中 sudo 报警数量，插件 id 和服务对应关系，参考第 1 章表 1-8
	reference_system	SELECT * FROM `reference_system` LIMIT 0 , 30	查询参考网址记录
alienvault_api	monitor_data	SELECT * FROM `monitor_data` LIMIT 0 , 30	查询主机 cpu、disk 等信息

3.4 MySQL 基本操作

大家在维护 OSSIM 时经常需要对 MySQL 进行操作，下面就是笔者总结出来的几个常见操作方法，数据库基本操作如下：

- 列出所有数据库：show databases
- 选择数据库：use databasename
- 列出表：show tables
- 显示表结构：show columns from tablesname
- 显示表字段完整属性：show full fields from tablesname
- 删除 student_course 数据库中的 abc 数据表：rm –f student_course/abc.*
- 查看数据库 alienvault 中 event 表的索引：show index from event
- 显示一个用户的权限：show grants for user_name
- 显示系统变量名称和值：show variables
- 显示系统正在运行的所有进程：show processlist
- 显示当前使用数据库的每个表信息：show table status
- 显示服务器所支持的权限信息：show privileges
- 显示安装以后可用的存储引擎和默认引擎：show engines
- 仅显示数据库、表、列信息：mysqlshow -uroot –puh4A73WQSr PCI

其中“uh4A73WQSr”为管理员密码，“PCI”为 OSSIM 库名称。

查看数据库 alienvault 的表空间（以 MB 为单位），要查找这类信息，可以找到 MySQL 中的 information_schema 表，这张表记录了所有数据库中表的信息，主要字段含义如下：

- TABLE_SCHEMA：数据库名
- TABLE_NAME：表名
- ENGINE：所使用的存储引擎
- TABLES_ROWS：记录数
- DATA_LENGTH：数据大小
- INDEX_LENGTH：索引大小

有了以上这些基础知识，我们开始稍复杂的几项操作：

（1）查看指定库的索引大小，例如数据库 alienvault 索引大小（以 MB 位单位），如图 3-14 所示。

```
mysql> select concat(round(sum(index_length)/(1024*1024),2),'MB') AS 'Total Inde
x Size' from information_schema.tables where table_schema like 'alienvault';
+------------------+
| Total Index Size |
+------------------+
| 74.24MB          |
+------------------+
1 row in set (0.76 sec)
```

图 3-14　查看 alienvault 库索引大小

（2）查看指定库（alienvault_api）的索引详细情况（以 GB 位单位），如图 3-15 所示。

```
mysql> select concat(table_schema,'.',table_name) AS 'Table Name',concat(round(t
able_rows/1000000,4),'M') AS 'Number of Rows', concat(round(data_length/(1024*10
24*1024),4),'G') AS 'Data Size', concat(round(index_length/(1024*1024*1024),4),'
G') AS 'Index Size',concat(round((data_length+index_length)/(1024*1024*1024),4),
'G') AS 'Total' from information_schema.tables where table_schema like 'alienvau
lt_api';
+---------------------------------------+----------------+-----------+-----------
-+---------+
| Table Name                            | Number of Rows | Data Size | Index Size
 | Total   |
+---------------------------------------+----------------+-----------+-----------
-+---------+
| alienvault_api.celery_job             | 0.0001M        | 0.0000G   | 0.0000G
 | 0.0000G |
| alienvault_api.current_status         | 0.0000M        | 0.0000G   | 0.0000G
 | 0.0000G |
```

图 3-15　查看指定库索引

3.5 OSSIM 系统迁移

随着部署 OSSIM 的深入，迁移系统的方法需要管理员掌握，有时我们需要将虚拟机上的 OSSIM 迁移到物理机，有时需要将故障 OSSIM 机器的数据迁移到新安装的机器……这些情况下都需要我们掌握迁移技巧。下文描述的方法就是假设有台发生故障的 OSSIM 服务器，需要迁移数据到新装的机器上的场景。

3.5.1　迁移准备

首先在故障机里保存重要信息：

（1）重要历史数据包括 events、alarms、assets、tickets，以及用户信息和生成的报表等。

对于 AlienvaultUSM 版而言这些重要数据存储在两个数据库中，MySQL 和 Mongo DB。

MySQL 储存 events、alarms、assets 信息，tickets 用户和权限，报表 MongoDB 存储 IDM 的历史数据。这些都非常重要。下面通过 SSH 远程登录到服务器进行备份。

MySQLDatabase Dump 操作：

```
alienvault:~# mysqldump -p `grep ^pass /etc/ossim/ossim_setup.conf | sed 's/pass=//'` --no-autocommit
--single-transaction --all-databases | gzip > alienvault-dbs.sql.gz
alienvault:~#
```

MonogoDatabasesDump 操作：

```
alienvault:~#mongodump -host localhost        \\生成 dump 文件
alienvault:~#tar cvzf alienvault-mongo.tgz dump   \\压缩 dump 文件
```

（2）环境：系统配置环境和插件等。

执行 backup.sh 脚本：

```
alienvault:~# cat backup.sh
#!/bin/bash
if [[ ! -f /etc/alienvault-center/alienvault-center-uuid ]];
then
dmidecode -s system-uuid |awk '{print tolower($0)}' >/etc/alienvault-center/alienvault-center-uuid;
fi
```

备份配置时间比较长，执行完以下命令将生成 alienvault-environment.tgz 文件：

```
alienvault-data.tgz  alienvault-dbs.sql.gz  alienvault-environment.tgz  backup.sh
alienvault:~# tar czvf alienvault-environment.tgz /etc/ossim/ /etc/alienvault/ /etc/alienvault-center/
 /etc/ansible/ /root/.ssh/ /home/avapi/ /home/avserver/ /var/ossec/ /etc/snort/ /etc/suricata/ /etc/na
gios3/ /etc/openvpn/ /var/cache/openvas/  /var/lib/openvas/ /etc/logrotate.d/ /etc/rsyslog.d/ /etc/apa
che2/ /usr/share/alienvault-center/ /etc/nfsen/ /etc/mysql/ /var/ossim/ssl/ /etc/hosts /etc/resolv.con
f
```

（3）裸数据（Raw Data）：包括 Logger、Netflow 以及重要抓包数据。备份命令如下：

```
Alienvault:~# tar czvf alienvault-data.tgz/var/ossim/logs/var/cache/nfdump/
```

不要尝试使用克隆系统（或 GHOST）的方式来解决迁移问题。

最后通过安全方式（WinSCP、SCP）将保存在服务器上的备份文件 alienvault-dbs.sql.gz、alienvault-mongo.tgz、alienvault-environment.tgz、alienvault-data.tgz 复制到目标 OSSIM Server 上。

3.5.2 恢复 OSSIM

当 OSSIM 的数据库意外受损，手动无法修复，此时最简单的方法可以通过 OSSIM 终端控制台进行数据库出厂设置，通过 ossim-setup 命令进入 AlienVault 控制台，菜单上选择 Maintenance&Troubleshooting→Maintain Database→Restore database to factory settings。恢复出厂设置是在 AlienVault 4.14.2 之后，新增加的一个实用功能，一旦进行数据库初始化恢复，那么再次登录 Web UI 就需要重新输入管理员密码，以及再次进行 2.5 节介绍的系统初始化工作，并要手动添加 Sensor，经过这几步复合操作之后就生成了全新的系统，不过读者需要明白手动恢复过程中的关键步骤：

（1）停止所有应用

```
/etc/init.d/monit stop
/etc/init.d/ossim-server stop
/etc/init.d/ossim-agent stop
/etc/init.d/ossim-framework stop
/etc/init.d/alienvault-idm stop
/etc/init.d/alienvault-center stop
/etc/init.d/alienvault-api stop
```

在上面第 5 步中，IDM 表示 Identiy　Management 身份认证管理。它将发现从同一个宿主事件发送过来的用户身份相匹配的信息。通过/etc/init.d/alienvault_idm start/stop 方式启动或停止 IDM，如果停止 IDM 服务，那么 SIEM 将无法填充安全事件。

（2）备份原 ossim_setup.conf 文件

```
#cp /etc/ossim/ossim_setup.conf  /root/ossim_setup.conf_last
```

（3）数据库恢复

```
#zcat alienvault-dbs.sql.gz |osism-db
#tar xvzf alienvault-mongo.tgz                              \\解压文件
#mongorestore  --db inventory dump/inventory/  \\恢复 MongoDB
```

（4）环境恢复

首先，删除当前配置。

```
#rm -rf /etc/ossim/ /etc/alienvault/ /etc/alienvault-center/ /etc/ansible/
/root/.ssh/ /home/avapi/ /home/avforw/ /home/avserver/ /var/ossec/ /etc/snort/
/etc/suricata/ /etc/nagios3/ /etc/openvpn/ /var/cache/openvas/
/var/lib/openvas/ /etc/logrotate.d/ /etc/rsyslog.d/ /etc/apache2/
/usr/share/alienvault-center/ /etc/nfsen/ /etc/mysql/ /var/ossim/keys/
/var/ossim/ssl/ /etc/hosts /etc/resolv.conf
```

然后，从备份中恢复。

```
#tar xvzf alienvault-environment.tgz -C /
```

接着，将 Alienvault 配置文件复制到/etc/ossim/目录下。

```
#cp /root/ossim_setup.conf_last /etc/ossim/
```

最后，执行下面命令更新文件权限。

```
#tar tvzf alienvault-environment.tgz | tr -s ' ' > file_list
#ulimit -s 65536
#cd /
```

```
alienvault:~# cat /root/permission.sh
#!/bin/bash
for i in `cat /root/file_list | cut -f2 -d " " |sort -u `;
do user=`echo $i | cut -f1 -d "/"`;
group=`echo $i |cut -f2 -d "/"`;
chown $user:$group `grep $i /root/file_list |cut -f6 -d " " |xargs`;
done
```

```
#ulimit -s 8192
#cd root
```

（5）RAW 数据恢复

解压 alienvault-data.tgz 到根目录。

```
#tar xvzf alienvault-data.tgz -C /
```

执行下面命令更新文件权限。

```
#tar tvzf alienvault-data.tgz | tr -s ' ' > file_list
#ulimit -s 65536
#cd /
```

```
alienvault:~# cat /root/permission.sh
#!/bin/bash
for i in `cat /root/file_list | cut -f2 -d " " |sort -u `;
do user=`echo $i | cut -f1 -d "/"`;
group=`echo $i |cut -f2 -d "/"`;
chown $user:$group `grep $i /root/file_list |cut -f6 -d " " |xargs`;
done
```

```
#ulimit -s 8192
```

（6）启动服务

```
#ossim-reconfig  -c -v -d
```

至此，将把故障 OSSIM 服务器的数据备份到新 OSSIM 服务器上并启动服务，由新机器接管原来的旧服务器。

3.6 OSSIM 数据库常见问题解答

1. 当 OSSIM 4 系统数据库发生损坏时，如何重建数据库

（1）手动清理，编辑/etc/ossim/ossim_setup.conf 文件，将其中 rebuild_database=no 改成 yes。

（2）运行如下命令：

```
#ossim-reconfig  -c
```

（3）再次将/etc/ossim/ossim_setup.conf 文件中 rebuild_database=yes 改成 no，完成自动方式，在命令行下运行：

```
#alienvault-reconfig --rebuild_db
```

2. 如何查询 OSSIM 数据库的 host 开头的表

```
#ossim-db
mysql>show tables like 'host%';
```

通过举一反三大家可以找到更多感兴趣的内容。

3. 如何备份 OSSIM 的 SIEM 数据库

OSSIM 4.6 以上系统的位置在 Configuration→Administration→Main→backup 下。在 Backup 栏里就可以按日期恢复。当然也可以手工选择使用 AutoMySQLBackup 工具（http://sourceforge.net/projects/automysqlbackup/）来备份，效果也不错。

4. 如何查看 MySQL 数据库信息

要查看 MySQL 数据库总大小、有多少个表，以及多少条数据库信息可以登录 phpMyAdmin，然后单击左侧的数据库，在右下方即可显示相关信息。在 MySQL 环境下使用如下命令：

```
SELECT sum(DATA_LENGTH)+sum(INDEX_LENGTH) FROM information_schema.TABLES
where TABLE_SCHEMA='数据库名'
```

另外，使用如下命令也能查看：

```
#du -h /var/lib/mysql
```

5. 如何查看 OSSIM 系统的 SIEM 数据库备份情况

SIEM 是其中非常重要的数据库，系统会备份 5 天，我们可以在 Configuration→Administration→Main→Backup 中查看备份策略，备份文件可以到/var/lib/ossim/backup 目录下查看。如果是远程备份的话，我们需要将这个目录下格式为 ossim-backup-日期.sql.gz 的文件，通过备份到远程服务器上妥善保存。在备份中有更多的细节大家可以参考/usr/share/ossim_framework/ossimframework/目录下的 Backup.py 脚本。

如果不小心单击 Configuration→Administration→Backup 下的“Clean SIEM Database”，系统这时会删除 SIEM 日志，在首页的仪表盘中会无法显示 Events by Sensor/Data Source 有关 SIEM 的信息。所以大家一定要备份好该日志文件，防止意外发生。而 Analysis→Alarm 信息是保存在 alienvault 数据库下的 alarm 表中，和 SIEM 日志无关，但我们也要学会如何保存。

6. 如何终止 OSSIM 数据库的僵尸进程

当遇到数据库的僵尸进程时，首先采用 show processlist 查看进程：

```
mysql> show processlist;
+-------+-----------------+-----------------+----------------+---------+-------+-----------------------+------------------+-----------+---------------+-----------+
| Id    | User            | Host            | db             | Command | Time  | State                 | Info             | Rows_sent | Rows_examined | Rows_read |
+-------+-----------------+-----------------+----------------+---------+-------+-----------------------+------------------+-----------+---------------+-----------+
|     1 | event_scheduler | localhost       | NULL           | Daemon  | 10253 | Waiting on empty queue | NULL            |         0 |             0 |         0 |
| 23664 | root            | localhost:47687 | alienvault     | Sleep   |     9 |                       | NULL             |         0 |             0 |         0 |
| 23665 | root            | localhost:47688 | alienvault_siem | Sleep  |  3208 |                       | NULL             |         0 |             0 |         0 |
| 23666 | root            | localhost:47689 | alienvault_siem | Sleep  |  3212 |                       | NULL             |         0 |             0 |         0 |
| 23676 | root            | localhost:47702 | alienvault     | Sleep   |  3203 |                       | NULL             |         1 |             1 |         1 |
| 23711 | root            | localhost:47763 | alienvault     | Sleep   |  3202 |                       | NULL             |        16 |           231 |       231 |
| 23714 | root            | localhost:47766 | alienvault     | Sleep   |   195 |                       | NULL             |         0 |             0 |         0 |
| 23715 | root            | localhost:47767 | alienvault     | Sleep   |   201 |                       | NULL             |         0 |             0 |         0 |
| 23717 | root            | localhost:47769 | alienvault     | Sleep   |   199 |                       | NULL             |         0 |             0 |         0 |
| 23718 | root            | localhost:47770 | alienvault     | Sleep   |   201 |                       | NULL             |         0 |             0 |         0 |
| 23720 | root            | localhost:47772 | alienvault     | Sleep   |   201 |                       | NULL             |         0 |             0 |         0 |
| 23742 | root            | localhost:47795 | alienvault     | Sleep   |    16 |                       | NULL             |         0 |             0 |         0 |
| 26558 | root            | localhost:51093 | alienvault     | Sleep   |  1628 |                       | NULL             |         0 |             0 |         0 |
| 78738 | root            | localhost:44271 | NULL           | Sleep   |   210 |                       | NULL             |         1 |             1 |         1 |
| 78750 | root            | localhost:44290 | alienvault     | Sleep   |   188 |                       | NULL             |         0 |             0 |         0 |
| 78976 | root            | localhost:44572 | alienvault     | Query   |     0 | NULL                  | show processlist |         0 |             0 |         0 |
+-------+-----------------+-----------------+----------------+---------+-------+-----------------------+------------------+-----------+---------------+-----------+
16 rows in set (0.00 sec)
```

然后用 kill 进程 id 的方法杀死进程。

7. 如果负载过大，在 OSSIM 系统中出现“MySQL :ERROR 1040:Too many connections”情况如何处理

如果出现以上报错信息，说明访问量比较高，一种方法就是用多个服务器分摊负载，另一种临时救急的方法是，修改 MySQL 配置文件/etc/mysql/my.conf 中的 max_connections 值，默认为 100 可以修改成 256。重启服务后，再继续观察。可以使用下面语句查看服务器响应的最大连接数：

```
mysql> show global status like 'Max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 26    |
+----------------------+-------+
1 row in set (0.00 sec)
```

若将 max_connections = 3000 或更大值，其实是无效的，MySQL 最大支持 1024。

新装的 OSSIM 系统默认只有 1024，当负载大时，我们需要修改配置，在/etc/security/limits.conf 文件中设置最大打开文件数，然后添加以下两行：

```
root soft nofile 65535
root hard nofile 65535
```

8. 如何远程导出 OSSIM 数据库表结构

命令行下具体用法如下：

```
mysqldump -u 用户名 -p 密码 -d 数据库名 表名 > 脚本名;
```

举例：服务器 IP：192.168.150.100，客户机为 192.168.150.21。首先在 MySQL 数据库中设置权限：

```
mysql>grant all privileges on *.* to 'root'@'192.168.150.21' identified by
'a1234567b' with grant option;
mysql>flush privileges;
```

然后在客户端输入命令：

```
#mysqldump -h 192.168.150.100 -u root -pa1234567b alienvault >dump1.sql
```

如果只需要导出单个数据表结构而不用包含数据，输入以下命令：

```
#mysqldump -h 192.168.150.100 -u root -pa1234567b -d alienvault >dump2.sql
```

备份 alienvault 数据库下的 vuln_nessus_servers 表的数据结构和数据：

```
#mysqldump -h 192.168.150.100 -u root -pa1234567b alienvault
vuln_nessus_servers >dump3.sql
```

9. OSSIM 系统出现 acid 表错误时如何处理

在使用过程中，OSSIM 数据库中的表可能发生故障，尤其在非法关机之后这种故障的概率更大，如图 3-16 中展示了一种故障现象，下面讲讲如何处理。

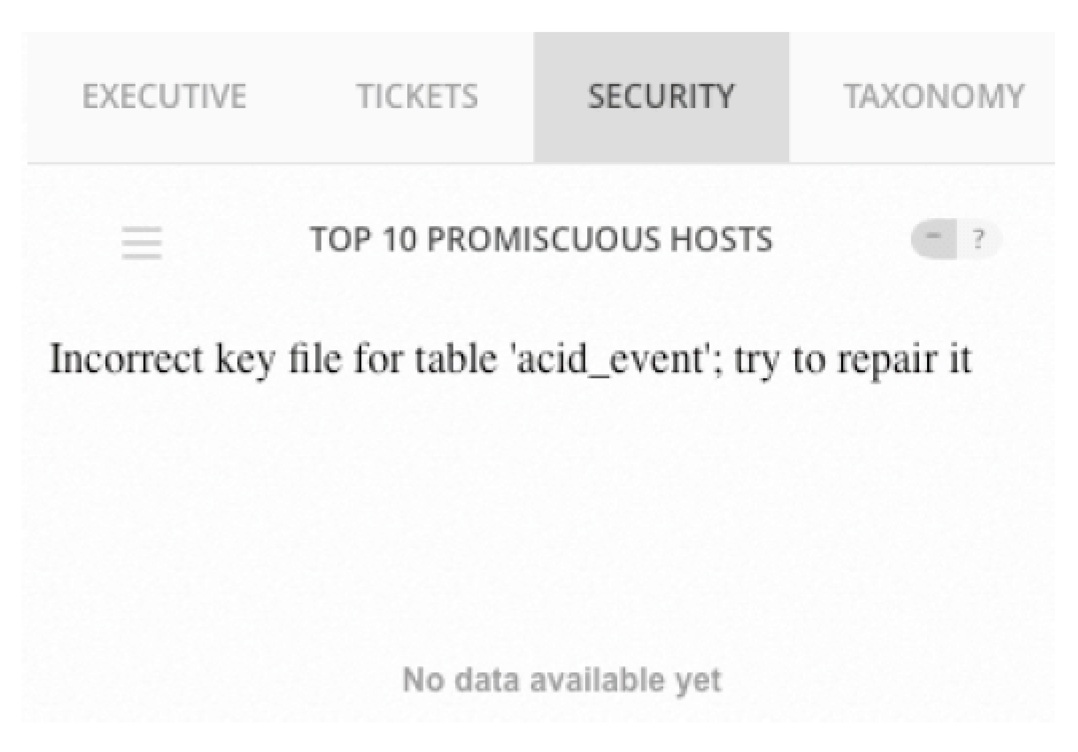


图 3-16　acid_event 表故障

首先，尝试使用 ossim-repair-tables 命令进行修复，如果无效，可以采用 MySQL 中的修复数据库工具 mysqlcheck，它可以不需要停止服务器来检查和修复表，方法如下：

首先需要知道数据库的 root 密码。

```
#cat /etc/ossim/ossim_setup.conf|grep pass
```

假设密码为“BiUGe4N5uD”，下面接着操作。

```
#mysqlcheck -o -u root -pBiUGe4N5uD -A      \\“-A”表示对所有库进行检查
```

10. 能否修改 OSSIM 系统中 MySQL 数据库密码

对于这个问题，初学者最好不要修改，因为密码是系统通过算法随机设置，相对较安全，最关键的问题是数据库密码在下列文件中都要调用：

```
/etc/apache2/conf.d/ocsinventory.conf
/etc/ocsinventory/dbconfig.inc.php
/etc/ossim/idm/config.xml
/etc/ossim/ossim_setup.conf
/etc/ossim/server/config.xml
/etc/ossim/agent/config.cfg
/etc/ossim/framework/ossim.conf
/etc/acidbase/databse.php
/etc/acidbase/base_conf.php
```

除非能将全部涉及的文件都修改完成，否则会出现无法连接数据库的情况发生。

11. 意外中断数据库写操作会对数据库的表造成损坏，如何检查表呢

有时候升级过程中系统会无法响应，可以使用到^Z 等命令终止升级，这时很有可能会对数据库的表造成损坏，为了检查/修复 MyISAM 表（.MYI 和.MYD），应该使用 myisamchk 实用程序。myisamchk 命令运行结果如图 3-17 所示。

```
#myisamchk -e *.MYI
```

```
alienvault:/var/lib/mysql/alienvault# myisamchk  -e *.MYI
Checking MyISAM file: repository.MYI
Data records:    1763   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
- check data record references index: 4
- check records and index references
alienvault:/var/lib/mysql/alienvault#
```

```
Checking MyISAM file: columns_priv.MYI
Data records:         0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1

---------

Checking MyISAM file: db.MYI
Data records:         2   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2

---------

Checking MyISAM file: event.MYI
Data records:         0   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
- check record links

---------

Checking MyISAM file: func.MYI
Data records:         9   Deleted blocks:       0
- check file-size
- check record delete-chain
- check key delete-chain
- check index reference
- check data record references index: 1
--More--
```

图 3-17　检查表

还有一种情况，在 Web UI 中遇到“*Checking for corrupt, not cleanly closed and upgrade needing tables.”错误提示，我们修复表之后方可排除错误。

12. 如何清理 OSSIM 数据库

很多情况下（例如分区不合理，数据量过大等情况），OSSIM 运行过一段时间，一些数据库会变得非常大，几十 GB 甚至更大，设置需删除数据库的一些内容以腾出空间，当用户删除数据时，MySQL 并不会回收，被已删除的数据占据的存储空间，以及索引位，都空在那里等待新的数据来弥补这个空缺，如果暂时没有数据来填补这个空缺，那就太浪费资源了，表面上看这个数据库文件的容量依然没有减少，对于写比较频繁的表，要定期进行 OPTIMIZE TABLE tablename。

清理数据库的几个步骤如下：

（1）停止 ossim-framework、ossim-server 以及 ossim-agent 服务。

（2）手动备份数据库。

```
>mysqldump (-u -h -p...) --all-databases > all_databases_2013-11-29_1.sql
```

（3）停止 MySQL 服务。

```
#service mysql stop
```

（4）删除现有数据库或者简单地移动它们也可以保存数据库文件。

```
#mkdir /mnt/disk1/var/lib/mysql-save/
#rsync -av /mnt/disk1/var/lib/mysql/ /mnt/disk1/var/lib/mysql-save/mysql
```

（5）开始 MySQL 服务。

```
#service mysql start
#ossim-db
  drop database ISO27001An;
  drop database PCI;
  drop database alienvault;
  drop database alienvault_asec;
  drop database alienvault_siem;
  drop database avcenter;
  drop database categorization;
  drop database datawarehouse;
  drop database myadmin;
  drop database mysql;
  drop database ocsweb;
  drop database osvdb;
  drop database performance_schema;
#service mysql stop
#service mysql start
```

（6）从备份中创建新数据库文件。

```
 #ossim-db < all_databases_2013-11-29_1.sql
```

（7）重启系统。

```
#service monit stop
#service ossim-agent stop
#service ossim-framework stop
#service ossim-server stop
```

在 OSSIM 中如果没有 nrpe 文件，需要使用 apt-install nagios-nrpe*来安装，然后在/usr/sbin/nrgp 中找到。

13. 如何用 xtrabackup 备份 OSSIM 数据库

Xtrabackup 是 percona 公司的开源项目，用来实现类似 innodb 官方的热备份工具 InnoDB Hot Backup 的功能，能够迅速地备份与恢复 MySQL 数据库。Xtrabackup 中包含两个工具：

（1）xtrabackup 是用于热备份 innodb、xtradb 表中的数据工具，不能备份其他类型的表，也不能备份数据表结构。

（2）innobackupex 是将 xtrabackup 进行封装的 Perl 脚本，提供了备份 myisam 表的能力。

由于 innobackupex 的功能更为全面和完善，故本文以 innobackupex 作为基础进行描述。下面介绍 xtrabackup 的全备份、增量备份与恢复的方法。用 percona xtrabackup 备份 OSSIM 数据库，安装 xtrabackup 过程如图 3-18 所示。

```
localhost:~# ls
README.txt            mysqlreport-3.5
alien_backup.sql      mysqlreport-3.5.tgz
alien_backup_full.sql percona-xtrabackup-dbg_2.1.4-657-1.squeeze_i386.deb
dump.sql              percona-xtrabackup-dbg_2.1.6-702-1.squeeze_i386.deb
innotop-1.9.1         percona-xtrabackup-test_2.1.6-702-1.squeeze_i386.deb
innotop-1.9.1.tar.gz  percona-xtrabackup_2.1.4-657-1.squeeze_amd64.deb
mbox                  tuning-primer.sh
localhost:~# dpkg -i percona-xtrabackup-test_2.1.6-702-1.squeeze_i386.deb
Running /usr/bin/dpkg -i percona-xtrabackup-test_2.1.6-702-1.squeeze_i386.deb
dpkg: error processing percona-xtrabackup-test_2.1.6-702-1.squeeze_i386.deb (--install):
 package architecture (i386) does not match system (amd64)
Errors were encountered while processing:
 percona-xtrabackup-test_2.1.6-702-1.squeeze_i386.deb
localhost:~# dpkg -i percona-xtrabackup_2.1.4-657-1.squeeze_amd64.deb
Running /usr/bin/dpkg -i percona-xtrabackup_2.1.4-657-1.squeeze_amd64.deb
Selecting previously deselected package percona-xtrabackup.
(Reading database ... 62517 files and directories currently installed.)
Unpacking percona-xtrabackup (from percona-xtrabackup_2.1.4-657-1.squeeze_amd64.deb) ...
Setting up percona-xtrabackup (2.1.4-657-1.squeeze) ...
localhost:~# dpkg -i percona-xtrabackup_2.1.4-657-1.squeeze_amd64.deb
```

图 3-18　安装 xtrabackup

首先，开始初始化配置，如图 3-19 所示。

使用如下命令：

```
innobackupex  --user=root  --password=xxxxxx /backup
```

```
localhost:~# mkdir /backup
localhost:~# innobackupex --defaults-file=/etc/mysql/my.cnf --no-timestamp --host=localhost --port=3
306 --user=root --password=jITg1VxAQB /backup

InnoDB Backup Utility v1.5.1-xtrabackup; Copyright 2003, 2009 Innobase Oy
and Percona Ireland Ltd 2009-2012.  All Rights Reserved.

This software is published under
the GNU GENERAL PUBLIC LICENSE Version 2, June 1991.

140119 09:38:16  innobackupex: Connecting to MySQL server with DSN 'dbi:mysql:;mysql_read_default_fi
le=/etc/mysql/my.cnf;mysql_read_default_group=xtrabackup;host=localhost;port=3306' as 'root'  (using
 password: YES).
140119 09:38:16  innobackupex: Connected to MySQL server
IMPORTANT: Please check that the backup run completes successfully.
           At the end of a successful backup run innobackupex
           prints "completed OK!".

innobackupex: Using mysql server version 5.5.33-31.1

innobackupex: Error: Failed to create backup directory /backup: File exists at /usr/bin/innobackupex
 line 416.
localhost:~# cat /etc/ossim/ossim_setup.conf|grep pass
mailserver_relay_passwd="unconfigured"
pass=jITg1VxAQB
update_proxy_pass="disabled"
localhost:~# _
```

图 3-19　开始备份

经过一段时间等待，备份完成，如图 3-20 所示。

```
innobackupex: Backup created in directory '/backup/2014-01-19_09-44-14'
140119 09:44:35  innobackupex: Connection to database server closed
140119 09:44:35  innobackupex: completed OK!
localhost:~# innobackupex -user=root --password=jITg1VxAQB /backup/_
```

图 3-20　备份完成

我们的备份工作还没结束，下面需要做统一检查，如图 3-21 所示。

```
Applications  Places  System
Terminal
File  Edit  View  Terminal  Tabs  Help
Terminal                                   Terminal
d: 657)
xtrabackup: uses posix_fadvise().
xtrabackup: cd to /var/lib/mysql
xtrabackup: using the following InnoDB configuration:
xtrabackup:   innodb_data_home_dir = ./
xtrabackup:   innodb_data_file_path = ibdata1:10M:autoextend
xtrabackup:   innodb_log_group_home_dir = ./
xtrabackup:   innodb_log_files_in_group = 2
xtrabackup:   innodb_log_file_size = 5242880
>> log scanned up to (81463275)
[01] Copying ./ibdata1 to /tmp/ibdata1
>> log scanned up to (81463275)
>> log scanned up to (81463275)
>> log scanned up to (81463275)
>> log scanned up to (81463275)
>> log scanned up to (81463275)
[01]        ...done
>> log scanned up to (81463275)
>> log scanned up to (81463275)
>> log scanned up to (81463275)
xtrabackup: The latest check point (for incremental): '81463275'
xtrabackup: Stopping log copying thread.
server1:~/innotop-1.9.1# xtrabackup_55 --defaults-file=/etc/mysql/my.cnf --backup --target-dir=/tmp
```

图 3-21　备份数据库并检查

接下来，模拟数据丢失，我们先手动删除 alienvault 库，如图 3-22 所示。

```
localhost:/var/ossim# cd /var/lib/mysql/
localhost:/var/lib/mysql# ls
ISO27001An      alienvault_asec   datawarehouse    ibdata1         mysql_upgrade_info
PCI             alienvault_siem   debian-5.5.flag  localhost.err   ocsweb
alienvault      avcenter          ib_logfile0      myadmin         performance_schema
alienvault_api  categorization    ib_logfile1      mysql
localhost:/var/lib/mysql# rm -rf alienvault
localhost:/var/lib/mysql#
```

图 3-22　删除 alienvault

打开 Web UI，然后开始恢复数据：

```
#innobackupex --copy-back /backup/2014-1-19/
```

记得将目录属性修改为 mysql.mysql：

```
chown mysql:mysql -R  <目录>。
```

在 OSSIM 可以直接使用 innotop 命令：

```
#wget http://innotop.googlecode.com/files/innotop-1.9.1.tar.gz
```

在 OSSIM 可以直接使用 innotop 命令。

14. 如何快速清除 SIEM 数据库

可在 Web UI 中选择 Configuration→Administration→Backup 下的 “Clear Siem Database”

按钮快速清除 SIEM 数据库。

15. 如何记录 OSSIM 数据库的执行过程

通过 MySQL 打开日志功能方法如下：

```
#vi /etc/mysq/my.cfg
```

在 Logging and Replication 选项（大约第 53 行的位置）下方，启用 log=/var/log/mysql/mysql.log。

保存文件后重启 MySQL 服务器即可，接着就可以查看日志了：

```
#tail -f /var/log/mysql/mysql.log
```

当打开 OSSIM Web UI 后，查询日志就会不停地增长，注意：这个功能会消耗系统性能，建议在负载较小时使用。

16. 如何优化表

alienvault.event 表数据比较大，随着更新和删除反复操作，数据在磁盘上的存储位置将会变得分布不均，同时会增加在该表中进行搜索的时间。在 MySQL 底层设计中，数据库将被映射到具有某种文件结构的目录中，而表则映射到文件，所以很有可能产生磁盘碎片，如果直接使用“optimize table event”命令，系统会提示“Table does not support optimize”，表示不支持优化，应采取以下方法优化，操作主要过程如图 3-23 所示。

```
mysql>alter table event ENGINE='InnoDB';
mysql>analyze table event;
```

```
mysql> analyze table event;
+------------------+---------+----------+----------+
| Table            | Op      | Msg_type | Msg_text |
+------------------+---------+----------+----------+
| alienvault.event | analyze | status   | OK       |
+------------------+---------+----------+----------+
1 row in set (0.00 sec)

mysql> optimize table event;
+------------------+----------+----------+------------------------------------------------------
-----------------------------+
| Table            | Op       | Msg_type | Msg_text
                             |
+------------------+----------+----------+------------------------------------------------------
-----------------------------+
| alienvault.event | optimize | note     | Table does not support optimize, doin
g recreate + analyze instead |
| alienvault.event | optimize | status   | OK
                             |
+------------------+----------+----------+------------------------------------------------------
-----------------------------+
2 rows in set (0.01 sec)
```

图 3-23　优化表

17. 如何使用 mysqldump 备份数据库

首先确保有足够磁盘空间，然后开始完全备份，如图 3-24 所示。

```
#mysqldump -u root -ppassword --all-databases >/root/mydump-2015-2-10.sql
```

```
localhost:~# mysqldump -u root -pyheZVRWgz8 --add-drop-database --add-drop-table --lock-all-tables -
-events --routines --triggers --databases ISO27001An PCI alienvault alienvault_api alienvault_asec a
lienvault_siem categorization datawarehouse myadmin ocsweb > alienvault_dump.sql
localhost:~# ll
total 313084
-rw-r--r-- 1 root root 320263525 Feb 20 03:27 alienvault_dump.sql
-rwxrwxrwx 1 root root       160 Jan  4 22:44 ping.sh
-rwxrwxrwx 1 root root       127 Jan  4 23:22 ssh.sh
-rwxrwxrwx 1 root root       187 Jan  4 23:38 test.sh
```

图 3-24　用 mysqldump 备份数据库

下面，开始重建数据库：

```
#alienvault-reconfig  -c  --rebuild_db
```

alienvault-reconfig 是一条命令，它等同于 ossim-reconfig 命令，后面"-c"、"--rebuild_db"为参数，之间用空格隔开，这个脚本执行过程比较复杂。首先它会备份 ossim-setup.conf 配置文件，更新/etc/issue、/etc/motd.tail、更新 Framework Profile、更新 Cron 文件、配置 Sensor Profile、更新 OSSEC 插件的 reference。完成这一系列动作之后，重启 ossim-server、squid3、Nagios3、nfsen、ossim-agent 服务，直到最后配置完成。下一步导入备份，备份命令例子如图 3-25 所示。

```
#mysql -u root -ppassword < /root/mydump-2015-2-10.sql
```

```
You have new mail in /var/mail/root
localhost:/tmp# mysqldump -u root -pyheZVRWgz8 --all-databases < /tmp/123.sql  &_
```

图 3-25　备份命令例子

改进方法：

```
VirtualUSMAllInOne:~# mysqldump -p`grep ^pass /etc/ossim/ossim_setup.conf | sed 's/pass=//'` --no-autocommit --single-tra
nsaction --databases alienvault | gzip > alienvault.sql.gz
VirtualUSMAllInOne:~# ll
total 1339648
-rw-r--r-- 1 root root  41227195 Feb 21 04:54 alienvault.sql.gz
-rw-r--r-- 1 root root 664471111 Feb 21 03:24 alienvault_dump.sql
-rw-r--r-- 1 root root         0 Feb 21 03:39 alienvault_dump.sqlcv
-rw-r--r-- 1 root root 664275825 Feb 21 03:39 alienvault_dump.sqls
-rw-r--r-- 1 root root      8065 Feb 20 21:59 pci-toplogy
-rw-r--r-- 1 root root   1801988 Nov 19  2012 percona-toolkit_2.1.7_all.deb
```

grep 后面的"^"代表首行匹配。

恢复方法：执行命令 zcat alienvault_dump.sql.gz |ossim-db，其操作过程如图 3-26 所示。

```
localhost:/tmp# zcat alienvault_dump.sql.gz | ossim-db
You have new mail in /var/mail/root
localhost:/tmp#
```

图 3-26　恢复备份

3.7　小结

本章首先分析了 OSSIM 数据库的结构以及作用，详细讲解了本地和远程主机访问数据库的注意事项，以及讲解数据库性能分析工具和监控、优化及迁移的方法。最后讲解了维护 OSSIM 数据库的常见问题。

第 4 章

OSSIM关联分析技术

从本章节可以学习到：

- 关联分析
- 不同种类日志的关联分析
- OSSIM 关联引擎
- 事件聚合及举例
- 海量事件处理
- 关联分析分类
- 通用关联检测规则
- 新建 OSSIM 关联指令
- OSSIM 自定义策略

4.1 关联分析技术背景

企业对自身网络安全日益重视，入侵检测系统（IDS）、防火墙（Firewall）、VPN 等网络安全设备被部署在企业网络中，但它们的作用未达到预期，由于技术实力存在不足，所以难以很好地解决网络安全问题，同时还会带来新的问题。

4.1.1 当前的挑战

许多安全管理者抱怨，已经设置了防火墙、入侵检测、防病毒系统、网管软件，为什么网络安全管理仍很麻烦。当前网络安全管理者面临如下挑战：

（1）安全设备和网络应用产生的安全事件数量巨大，IDS 误报严重。一台 IDS 系统，一天产生的安全事件数量成千上万，通常 99%的安全事件属于误报，而少量真正存在威胁的安全事件淹没在误报信息中，难以识别。

（2）安全事件之间存在的横向和纵向方面（如不同空间来源、时间序列等）的关系未能得到综合分析，因此漏报严重，不能实现实时预测。一个攻击活动之后常常接着另一个攻击活动，前一个攻击活动为后者提供基本条件；一个攻击活动在多个安全设备上产生了安全事件；

多个不同来源的安全事件其实是一次协作攻击，这些都缺乏有效的综合分析。

（3）安全管理者缺乏对整个网络安全态势的全局实时感知能力。

4.1.2　基本概念

充分利用多种安全设备的检测能力，集中处理的致命弱点是待分析处理的数据量巨大，那些庞大冗余，独立分散，安全事件显然不能直接作为响应依据，同时网络安全防护也有实时性要求，上述问题的根本解决途径是网络安全事件关联处理。到底什么是关联分析呢，有几个基本的概念大家需要了解。

（1）安全事件：第 1 章开始就提到了安全事件，包括服务器安全日志，重要应用的告警以及日志。

（2）数据源（Date Source）：数据源是安全事件的来源，这里包括防火墙、入侵检测系统，重要主机、路由交换设备的日志。

（3）数据关联：将多个数据源的数据进行联合（Association）、相关（Correlation）或组合（Combination）分析，以获得高质量的信息。它将不同空间设备的日志，不同时间序列存在的问题经过特定关联方法结合在一起，最终确定工具的分析方法。当然这些只是广义的安全事件关联方法，后面章节还会专门针对 OSSIM 讲解具体的规则。

（4）交叉关联：它是最常见的数据关联方式，可以将安全事件与网络拓扑、系统开放的服务、设备存在的漏洞进行关联匹配，以分析攻击成功的可能性。利用这种关联方法可以在 OSSIM 系统中关联规则检测到某些威胁，并实现自动响应（比如发出告警等）。

下面举几个异常实例：

- 完整性方面。文件完整监控工具发现系统中 ls、ps、netstat、su 等程序大小和所有者被改变，可判断受到攻击。
- 系统方面。用户账号被修改及一些不能解释的异常登录行为，在不可能的地方出现了新的文件、目录或者丢失了文件、目录大小急速增大、骤然减小、MD5 签名不匹配，这些迹象都说明系统已遭受入侵。
- 日志方面。系统日志缩减，日志中出现了不明条目、异常的中断消息都说明了系统遭受入侵。
- 流量方面。若干节点流量大规模增大表示可能遭受拒绝服务攻击。

4.1.3　安全事件之间的关系

上述这些状况看似孤立，实际有着联系，因为攻击者表面上的单个攻击行为，可以在多个地方产生安全事件，比如攻击者从外网非法访问数据库，那么攻击者经过路径中的所有应用程序及安全设备都会记录这种异常行为的日志。所以网络中安全事件的关系大致分为：并列关系，时间序列关系（首先扫描到目标，然后根据目标开放端口进行针对扫描）和冗余关系（针对单个目标的多个端口或多个目标的多个端口的扫描行为，即会触发多个安全事件）。

4.2 关联分析基础

4.2.1 从海量数据到精准数据

根据 ISO 27001 的定义，事件关联分析就是指用户将海量安全事件，进行相关性分析，定位真正故障点的过程。那么海量数据并结合复杂类型就形成了大数据。下面讲到的实质就是大数据分析的一部分内容。

表 4-1 展示了一组攻击状态下的模式和每个模式所具有的特征集，事件关联分析就是要将这些模式和特征进行提炼，找出问题和内在联系。这种分析难度已超越了我们以前利用正则表达式过滤日志的难度，所有这些事件日志代表着扫描、攻击、漏洞利用或服务器故障的形式，而通过人工查询日志的方法很难找出关联问题。

表 4-1　模式与特征（√表示有，×表示无）

模 式	暴力破解登录	端口扫描	渗 透
大量出站流量	×	×	√
大量入站流量	×	×	×
非工作时间登录 VPN	×	√	√
防火墙 Access	√	√	√
防火墙 deny	×	×	×
从内网之外登录	×	×	√
连续多次登录失败	√	×	×
至少一次登录成功	√	×	×
单一来源探测多个目标 IP	×	√	√
单一来源探测多个 IP 及端口	×	√	√

OSSIM 的核心技术之一就是关联分析，通过这项技术能够挖掘出海量的安全日志中隐藏的信息，这样可以帮助安全人员发现入侵行为，本章将深入为读者剖析关联技术及应用。

下面举一个例子：一个拥有 500 台 PC 的中小型企业，其网络安全产品每天产生的事件估算数量，如表 4-2 所示。

表 4-2　企业日志产量统计

安全目标	安全产品	日志产量（/天）
网络内网安全	桌面管理系统	大于 1000 条
防病毒系统	防病毒服务器、防病毒网管	近 50000 条
网络安全	交换机、路由器	大于 1000 条
	IDS/IPS	大于 500000 条
	防火墙、堡垒机	大于 2000000 条
保护关键业务	主机审计、应用审计	大于 100000 条

如果简单地将这些日志全部收集起来，而不进行关联分析，对于潜在安全事件还是束手无策，对运维效率来说不升反降。因此，安全管理评估不仅要收集这些海量事件，还要通过关联分析引擎进行事件“提纯”，也就是输出很少量的、真正值得管理员关注的安全事件，如图 4-1 所示。

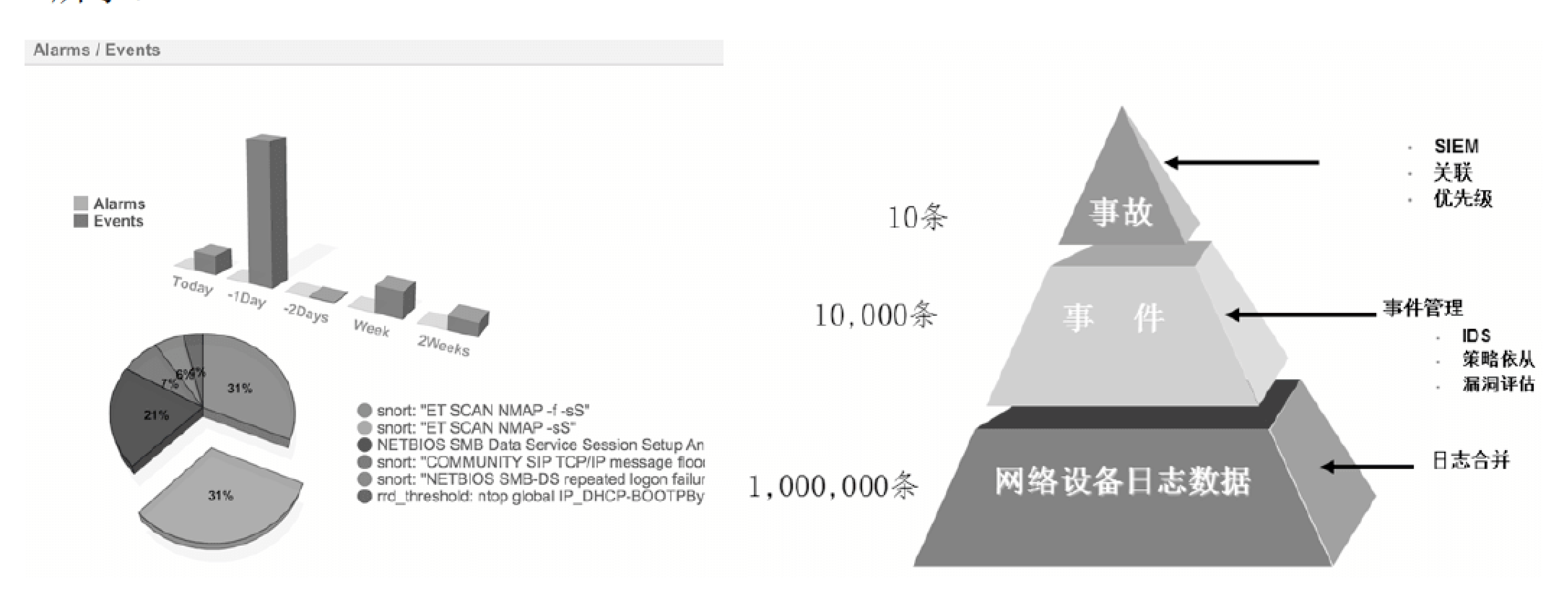


图 4-1　Alarm 的提取

从图 4-1 中可以看出 IDS 系统产生的 Alerts 和 Alarms，我们通过在 OSSIM 中，对比 SIEM 和 Alarm 的数值可知。同时读者也可观察 Dashboards→Overview→Executive 中图形化显示的 SIEM 和事件数量，他们和 Alarm 都不在一个数量级。同样在 SIEM :TOP 10 的事件分类饼图中，所有事件之和等于总 SIEM 的数量。对这些现象来说，在它们之间起到关键作用的就是关联引擎。

4.2.2　网络安全事件的分类

网络攻击具有多种表现形式及不同的危害，对不同攻击产生的安全事件进行分类，具有显著的意义。基于统计的分类方法，如根据安全事件相同源的数量、相同目的的数量、相同目的端口的数量等进行统计分类，这样就可看出应关注的一些源、目标、目标端口等，通过它们来发现异常。在 OSSIM 的 Report 报表菜单中（或者在 Analysis→Alarm 中的 Alarm Report 报表）可根据安全事件的源和目的 IP，得到十分详细的统计分类，如图 4-2、图 4-3、图 4-4 所示。

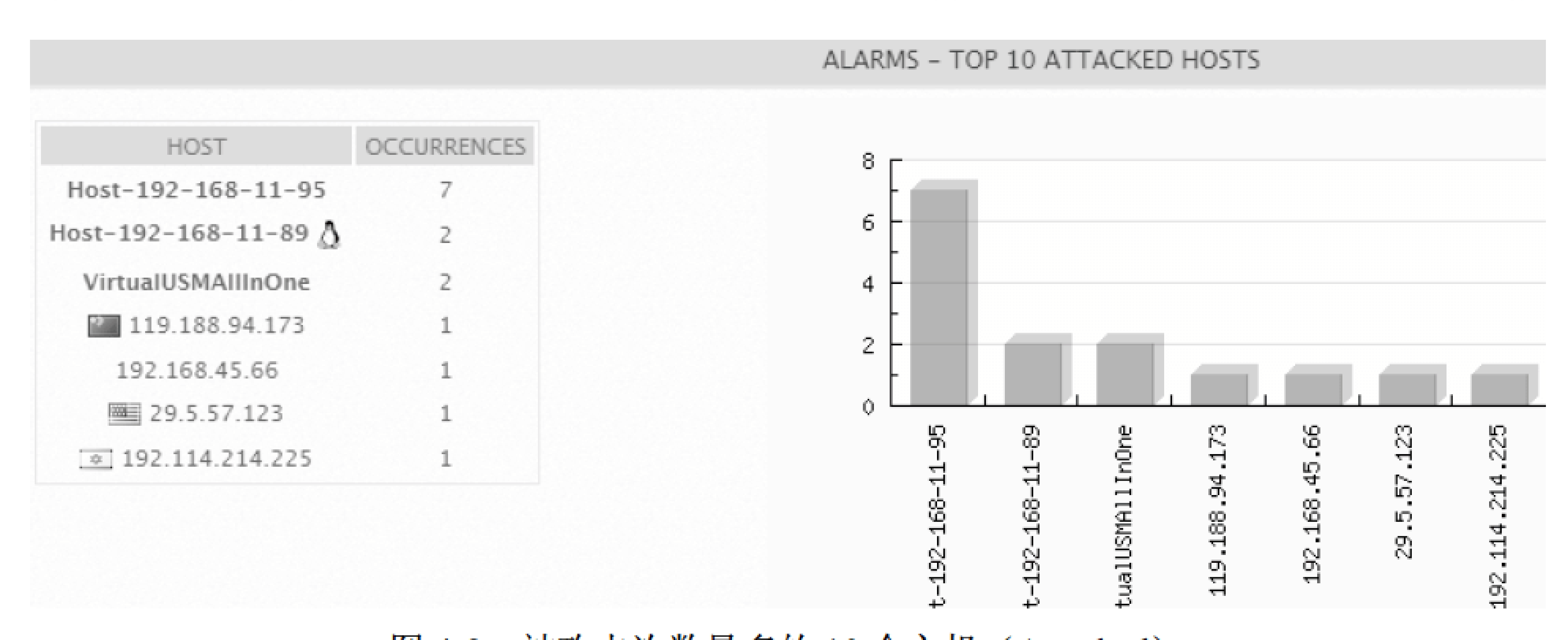


图 4-2　被攻击次数最多的 10 个主机（Attacked）

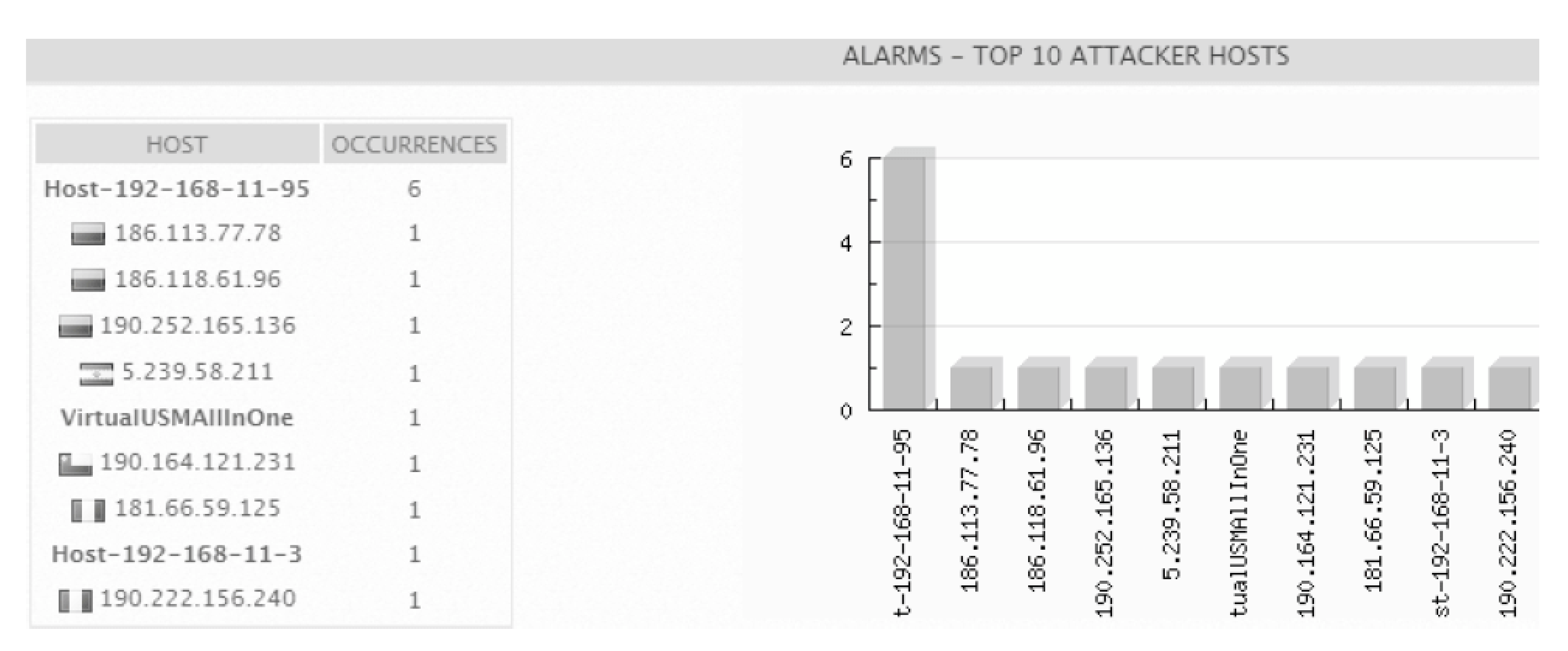


图 4-3 发出攻击最多的 10 个主机（Attacker）

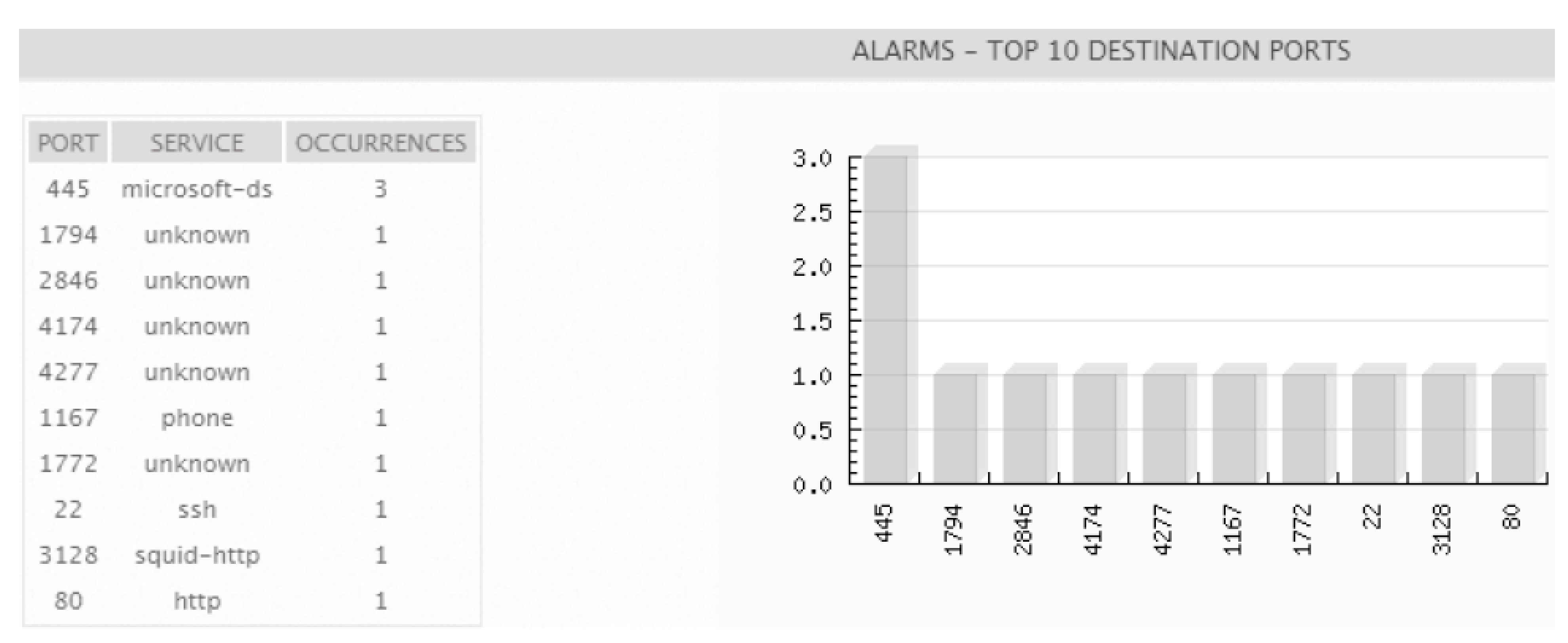


图 4-4 攻击的目标端口

在 OSSIM 系统中，所有原始日志收集到 Analysis→RAW Logs 下，通过柱状图展示出来，日志详情存储在/var/ossim/logs 目录下，经过归一化处理后的事件存放在 alienvault_siem 库的 acid_event 表中，展示在 Web UI 的 Dashboards→Overview→Executive，以及 Dashboards→Overview→Security 页面上。经过系统筛选出的事故，则存储在 alienvault 库的 alarm 表中，展示在 Analysis→Alarm 中。

在没有用到 OSSIM 之前，我们会通过类似 grep "Failed password for root" /var/log/auth.log | awk '{print $11}' | sort | uniq -c | sort -nr | more 的命令显示登录失败的统计信息，如果频繁出现问题，那么操作会变得比较烦琐。下面我们看看 OSSIM 是如何处理。我们抽取一条存储在/var/log/auth.log 中 Ssh 登录失败的原始日志，以及归一化处理后的事件进行对比，如图 4-5、图 4-6 所示。

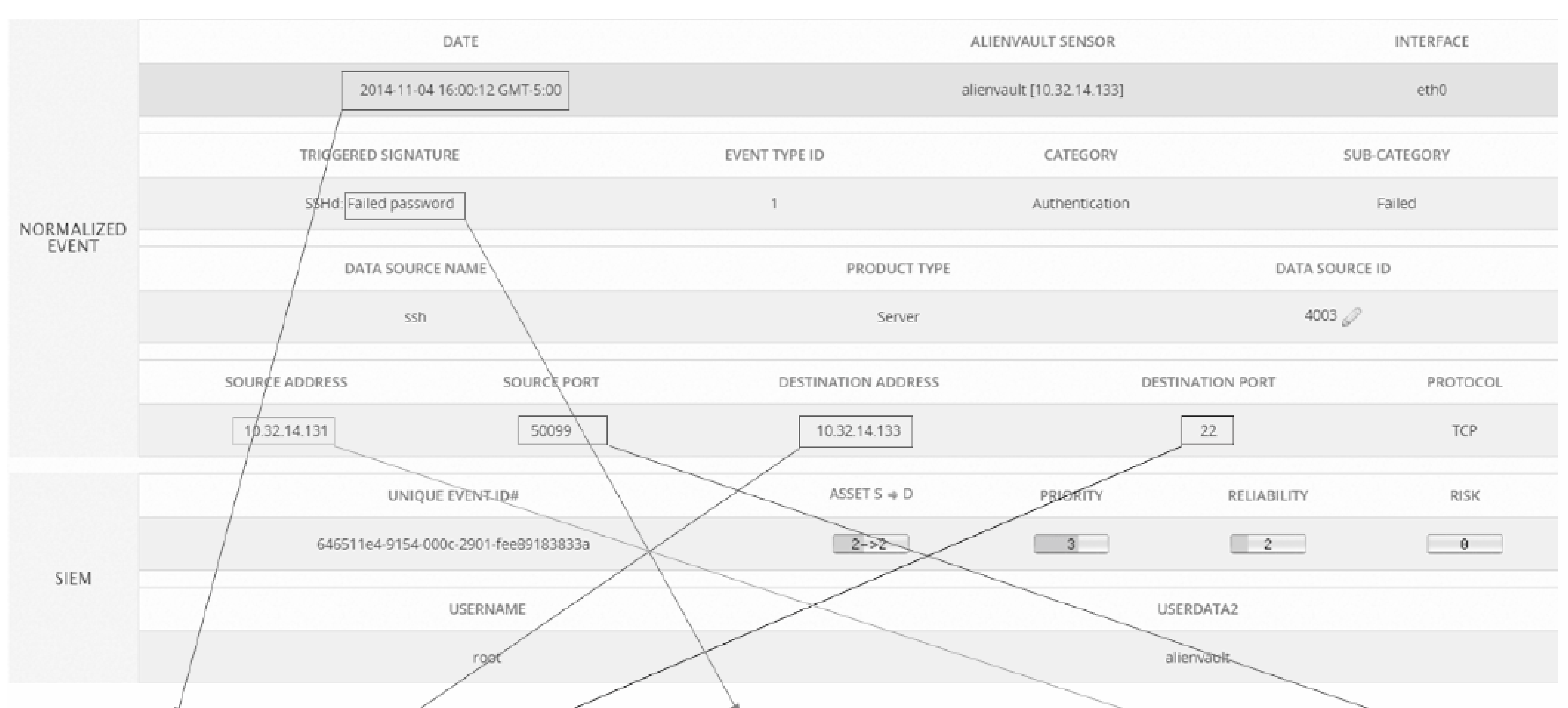


图 4-5　SSH 原始日志和归一化处理的对比

DASHBOARDS　ANALYSIS　ENVIRONMENT　REPORTS　CONFIGURATION

ALARMS
SECURITY EVENTS (SIEM)
RAW LOGS
TICKETS

7 2014-10-05 13:16:04 sshd 0.0.0.0:22 192.168.11.105 Oct 5 13:16:04 alienvault sshd[5194]: Failed password for root from 192.168.11.151 port 2510 ssh2 [...] Validate

8 2014-10-05 01:16:05 ossec authentication 0.0.0.0 192.168.11.105 AV - Alert - "1412486165" --> RID: "5716"; RL: "5"; RG: "syslog,sshd,authentication_failed,"; RC: "SSHD authentication failed."; USER: "None"; SRCIP: "192.168.11.151"; HOSTNAME: "alienvault"; LOCATION: [...] Validate

9 2014-10-05 13:16:04 sshd ossim411 192.168.11.151:2511 0.0.0.0:22 192.168.11.105 Oct 5 13:16:04 alienvault sshd[5196]: Failed password for root from 192.168.11.151 port 2511 ssh2 [...] Validate

10 2014-10-05 13:16:04 sshd ossim411 192.168.11.151:2512 0.0.0.0:22 192.168.11.105 Oct 5 13:16:04 alienvault sshd[5200]: Failed password for root from 192.168.11.151 port 2512 ssh2 [...] Validate

EVENT TYPE	EVENT DETAIL
sshd / SSHd: Failed password	Oct 5 13 16 04 alienvault sshd[5200] Failed password for root from 192.168.11.151 port 2512 ssh2

SENSOR	PRODUCT TYPE	CATEGORY	SUB-CATEGORY
ossim411	Server	Authentication	Failed

USERDATA1	USERDATA2	USERDATA3	USERDATA4	USERDATA5	USERDATA6	USERDATA7	USERDATA8	USERDATA9
empty	alienvault	empty	empty	empty	empty	empty	empty	empty

11 2014-10-05 13:16:04 sshd ossim411 192.168.11.151:2513 0.0.0.0:22 192.168.11.105 Oct 5 13:16:04 alienvault sshd[5203]: Failed password for root from 192.168.11.151 port 2513 ssh2 [...] Validate

12 2014-10-05 13:16:04 sshd ossim411 192.168.11.151:2515 0.0.0.0:22 192.168.11.105 Oct 5 13:16:04 alienvault sshd[5212]: Failed password for root from 192.168.11.151 port 2515 ssh2 [...] Validate

13 2014-10-05 13:16:06 sshd ossim411 192.168.11.151:2518 0.0.0.0:22 192.168.11.105 Oct 5 13:16:06 alienvault sshd[5221]: Failed password for root from 192.168.11.151 port 2518 ssh2 [...] Validate

图 4-6　RAW 日志

事件序列关联可以从诸多安全事件中，找出属于同类攻击意图的多次攻击活动（例如暴力破解）对应的安全事件，比如对于 SSH 的暴力破解，如图 4-7、图 4-8、图 4-9 所示。

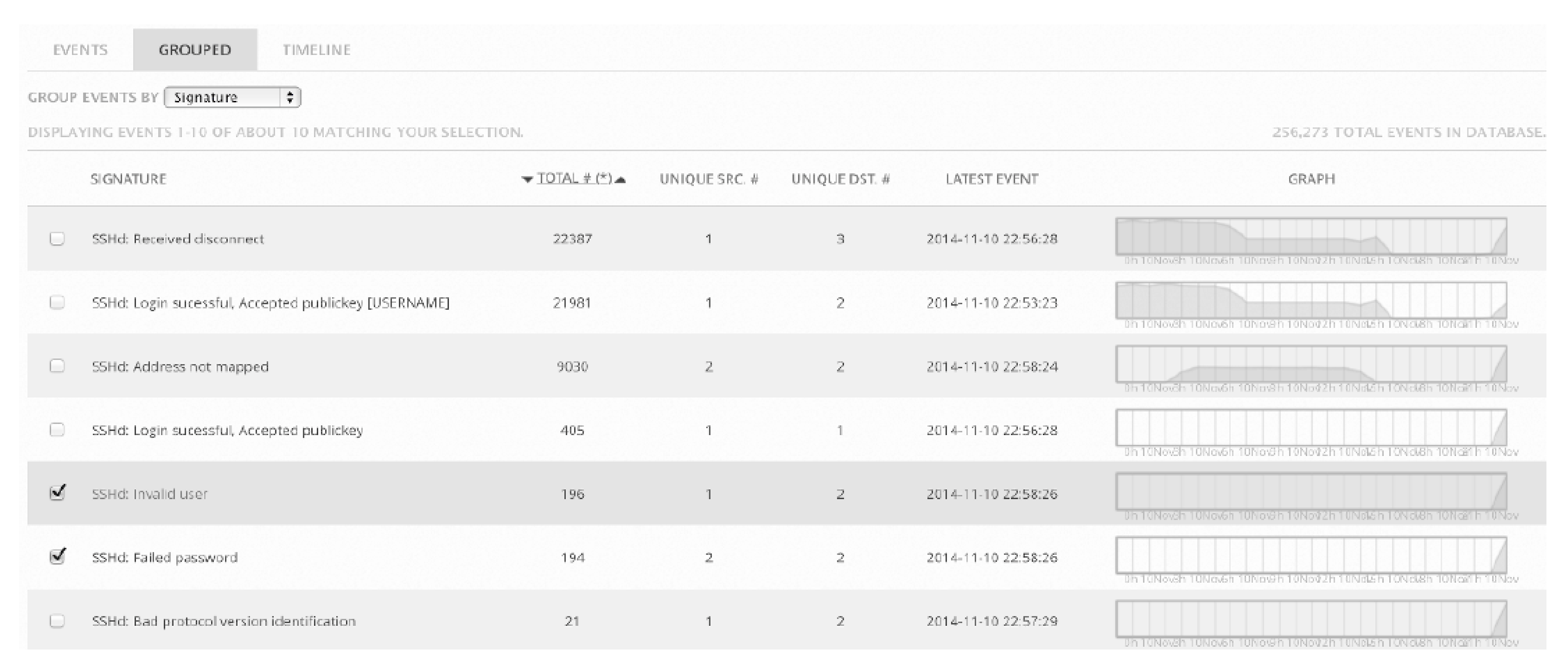


图 4-7　事件分组

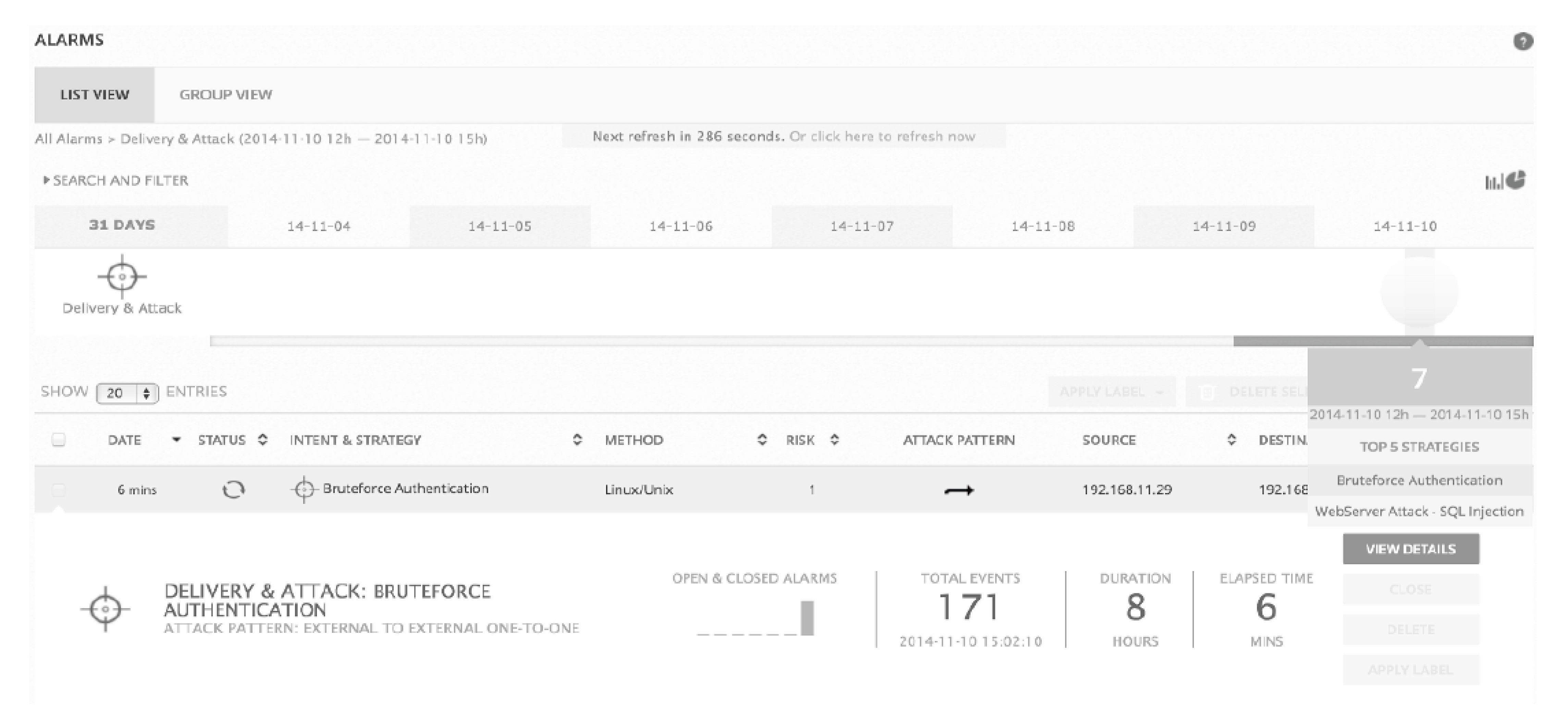


图 4-8　攻击类事件

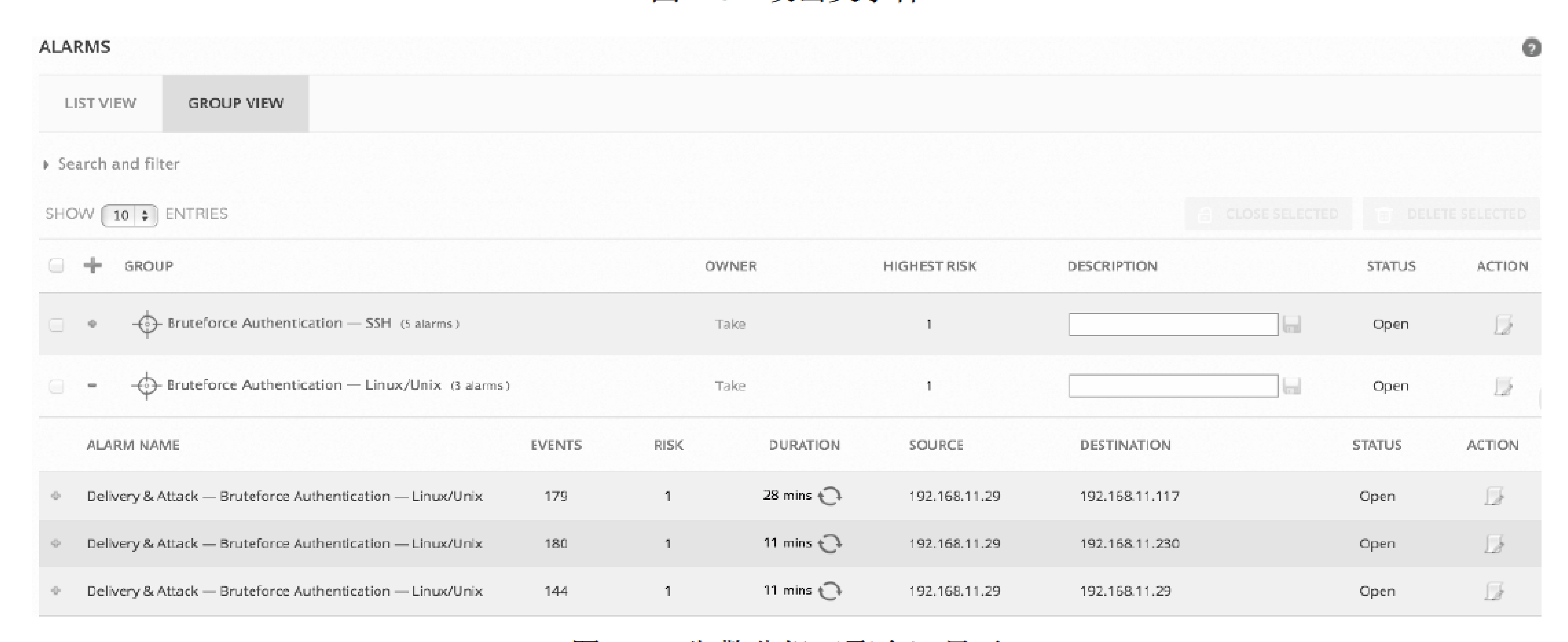


图 4-9　告警分组（聚合）显示

4.2.3　Alarm 与 Ticket 的区别

Ticket 用在行车中由于司机违规，由交警开出的罚单。Ticket 可视为一种更紧急的报警类型，系统发现威胁或严重漏洞后给管理员的提示，这里是指系统发现了严重漏洞并报给管理员。查看 Ticket 如图 4-10 所示。

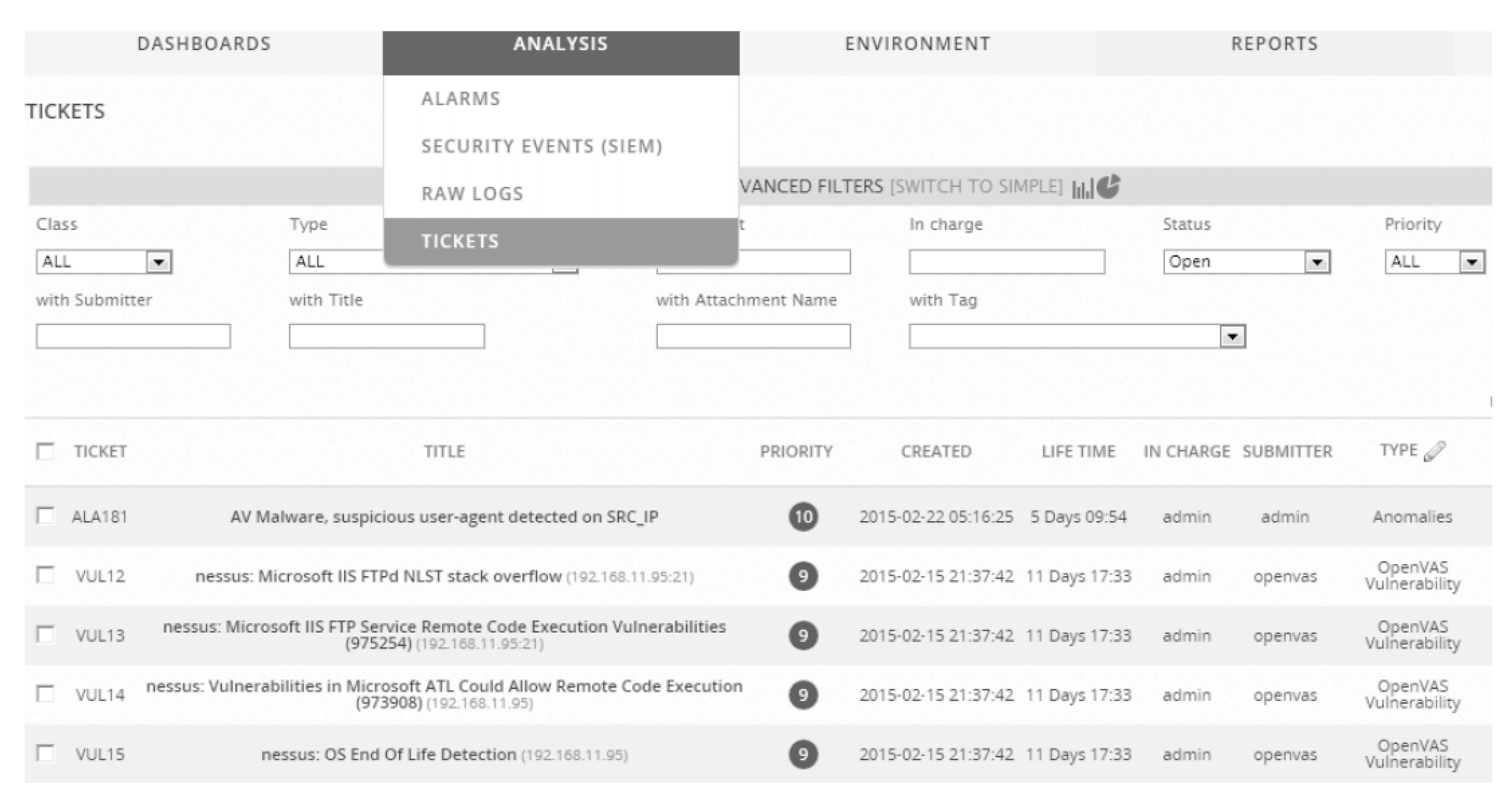


图 4-10　Tickets 显示

通常发生了 Ticket 它的优先级都较高（通常>5），可通过 Ticket 给管理员发送邮件提醒，其目的是在攻击者成功入侵网络之前，第一时间提醒问题发生了，促使管理员去锁定故障源。

前面讲过，Alerts>>Alarms（注意：符号“>>”表示远大于），我们再深入思考一下，OSSIM 系统在什么情况发会发出 Alarm？进过分析发现，当系统检测到事件的风险大于 1 时，将产生报警（alarm），那风险 Risk 的计算就显得很重要。而且在 SIEM 事件分析控制台中用 3 个级别来描述风险等级，RISK Low Low Medium High 分别是，低级别 Risk<1；中级别 Risk≥1，高级别 High ≥3。

从数量上看 Alerts>>Alarms>>Tickets，Ticket 是从众多 events、alarm 里提取的精华。alarm 只是提取一个事件，并根据风险程度来创建一个通用报警，以供查看。报警不会分配给系统用户，但 Ticket 可以分配给系统用户，例如 admin，还能给出更精细的控制，比如发送邮件给某个用户。将它们两个的联系可以放在一张图上进行对比，如图 4-11 所示，这是在 OSSIM Web UI 下仪表盘中的 Alarm 和 Tickets 的对比。

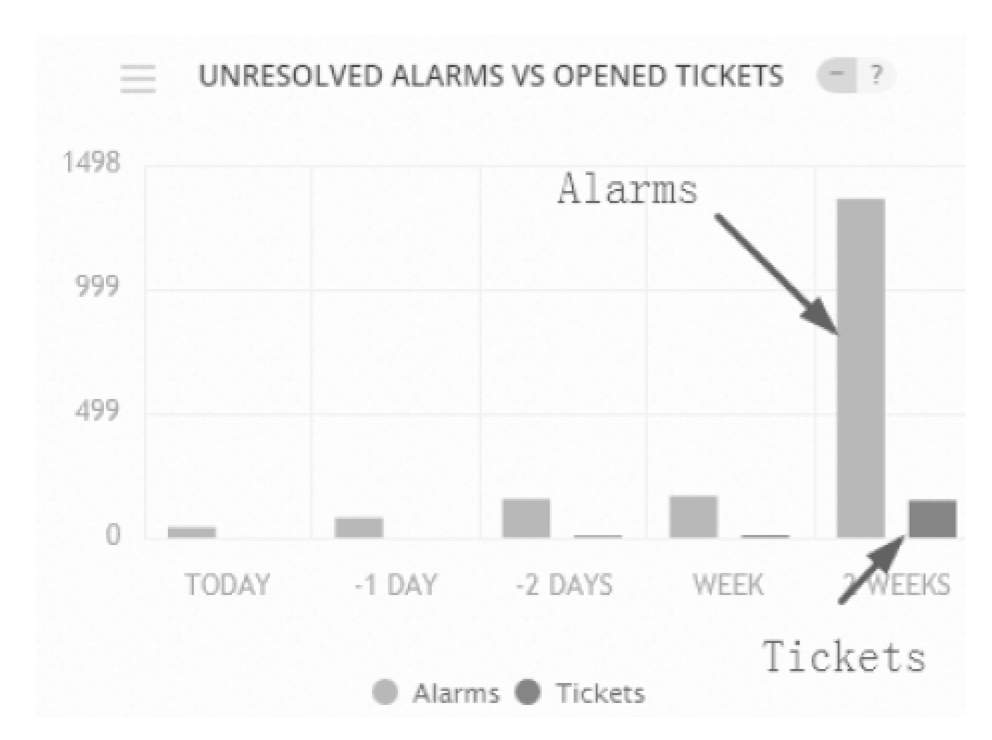


图 4-11　Alarm 与 Tickets 数量对比的柱状图

这里指的是对于严重的漏洞，系统需要创建一个票据系统，通常用 Tickets 表示，它的值是 1~9，这个值可以自己修改。如果改成“1”，那么在一次扫描中很可能获得成千上万条的 tickets。面对诸多信息，如何甄别？在实际工作中，发现设置为“7”或“8”比较适合。

4.2.4　使用 Ticket

在实践中发现，Ticket 除了系统识别的以外，我们还能手动添加新的 Tickets。在 SIEM 控制台上选中某个事件并添加，如图 4-12 所示。

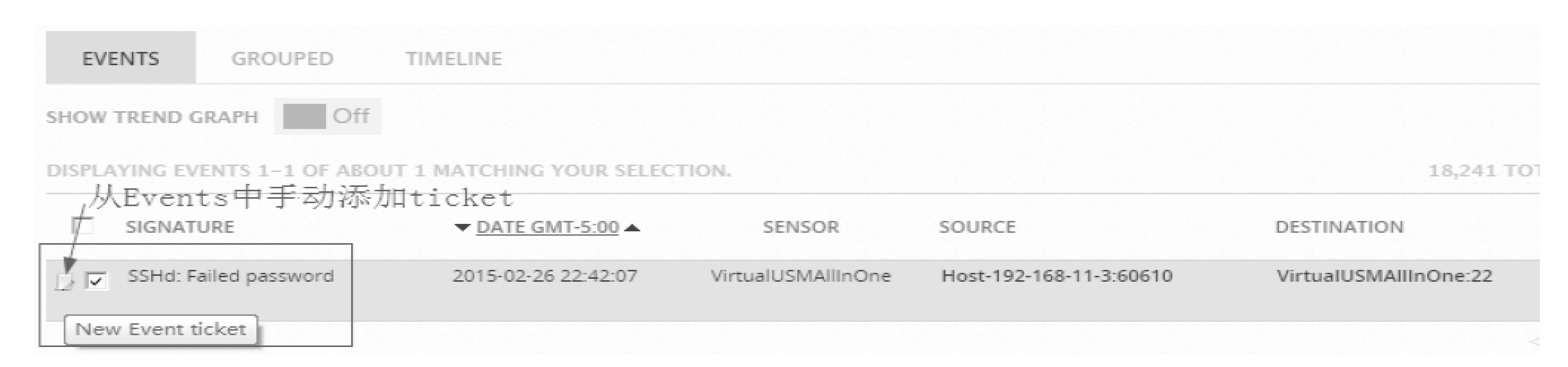


图 4-12　手动添加 Ticket

接下来弹出如图 4-13 所示的对话框，用户需要填写名称、优先级、类型等信息即可完成注册。

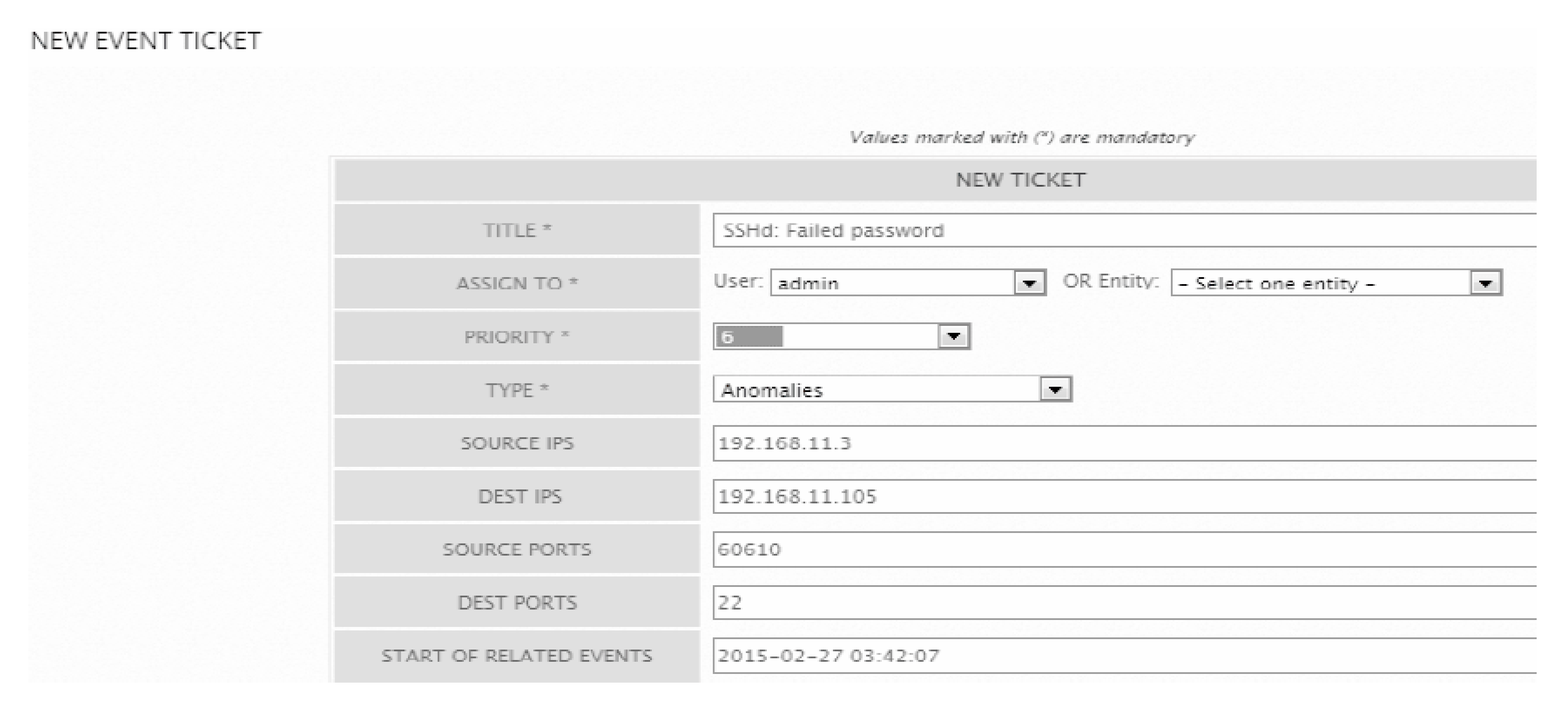


图 4-13　新建 Ticket 举例

接着，在 Tickets 中的筛选条件中选择 events 类（注意还包括 Vulnerability、Anomaly、Metric、Event、Alarm 等其他类），即可立刻将刚才建立的 tickets 展示出来，如图 4-14 所示，而且在 Dashboards→Overview→Tickets 中可以查看到各种以 Tickets 为主题的报表。

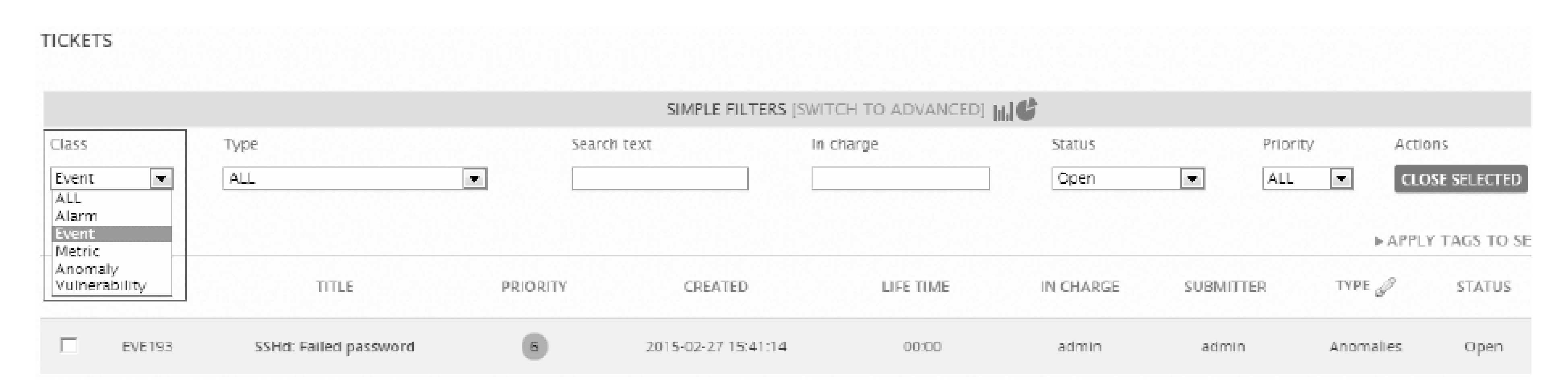


图 4-14　查看 Ticket

4.2.5　加入知识库

最后，我们可以将典型的 Tickets 新建文档加入到系统知识库中，知识库在企业漏洞管理中尤为重要，在默认的 OSSIM 系统中具有 1763 条目，可以根据自身情况添加知识库信息。还是以上面的事件为例，我们单击 Tickets 中的“SSHd Failld password”出现如下画面，再单击“NEW DOCUMENT”创建新文档，如图 4-15 所示。

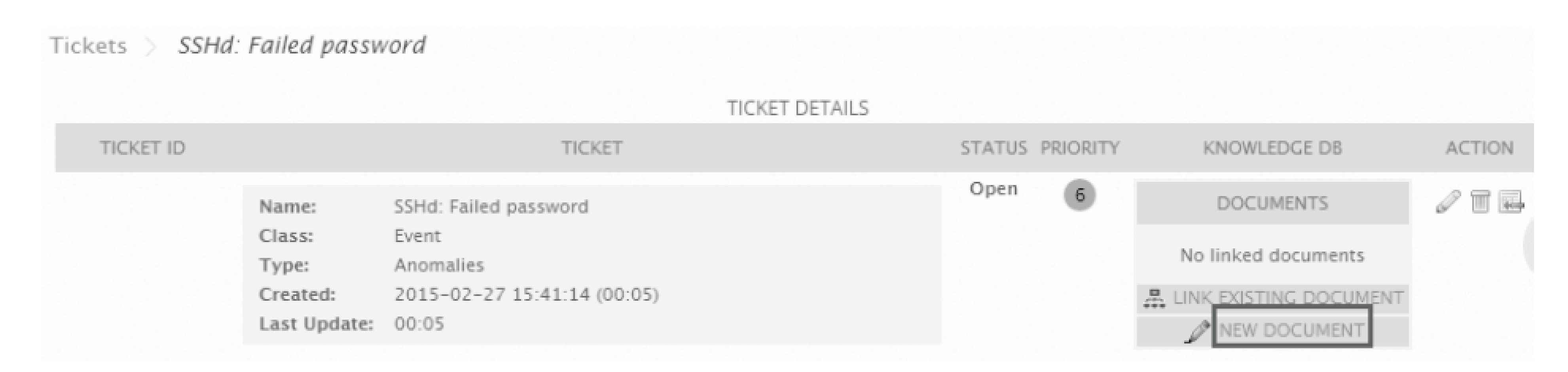


图 4-15　新建文档

打开对话框，如图 4-16 所示，根据一定语法规则加入描述信息，以及上传附件信息。注意：目前 KDB 还不支持中文。

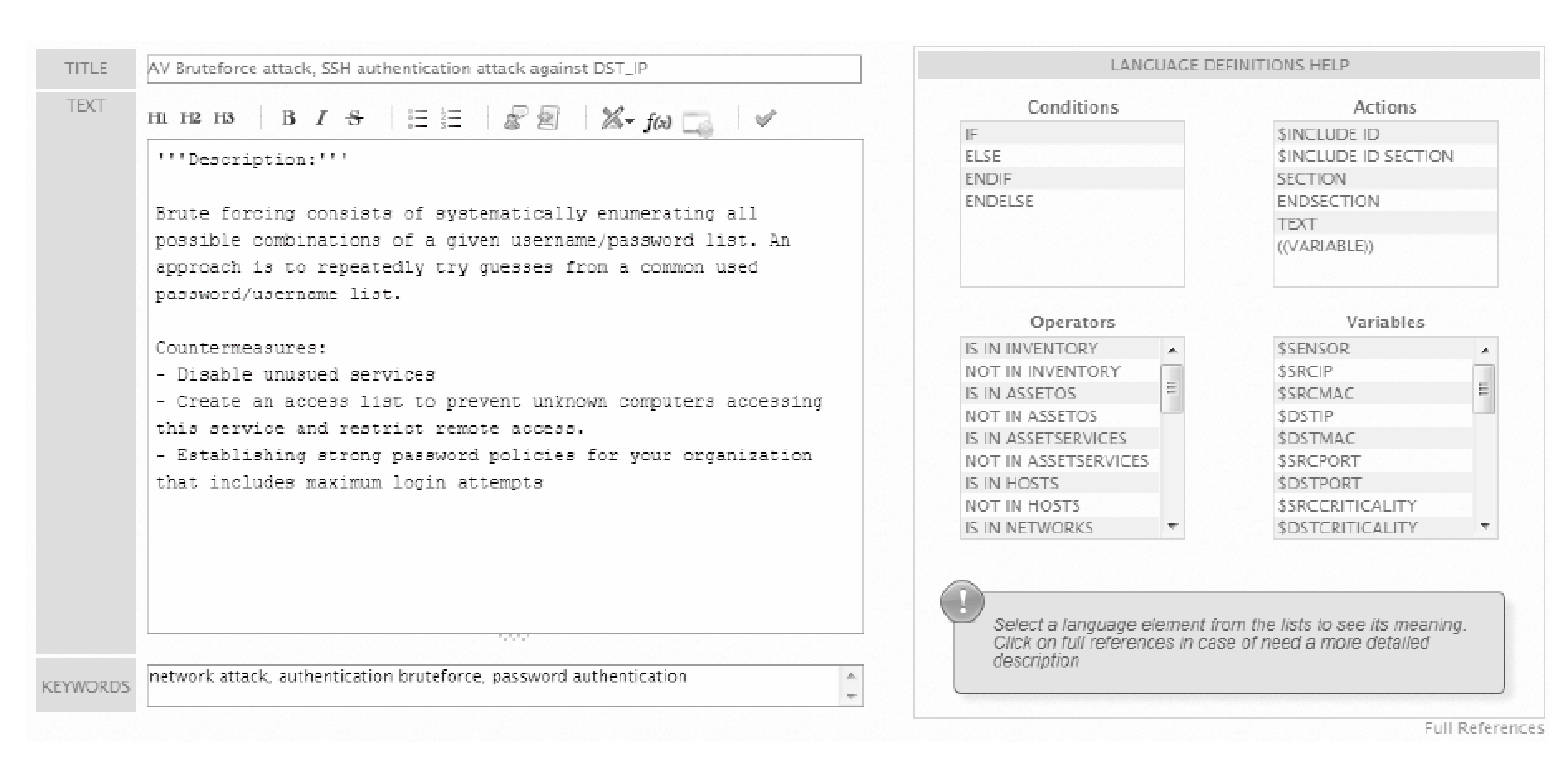


图 4-16　添加进 KDB

4.2.6 安全事件提取

如何在庞大日志中提取出真正人们感兴趣的事件，并最终形成安全事件告警呢？这就是事件的提取过程，从第 1 章介绍 OSSIM 结构得知，Sensor 中具有代理功能，主要负责日志采集并进行归一化处理，形成统一的安全事件格式，下面紧接着要做的处理就是由聚合模块消除冗余安全事件，再由交叉关联与序列分析等模块进行更深层次的关联。

下面我们再看几个关联分析场景举例：

（1）比如在 VPN 服务器日志显示张三 3:00 点钟，从外网登录到内部网，3:05 分登录 FTP 服务器，并在 FTP 服务器上下载了某一文件。在门禁系统的日志显示张三在不久前刚刚进入办公区域，这三个日志可以关联出一个安全事件。

（2）某公司核心数据库前端部署了防火墙系统，拓扑示意如图 4-17 所示，某日安全系统监测发现张三登录了 MySQL 数据库服务器，但是在防火墙日志中并没有发现张三的访问日志，则说明张三很有可能绕过防火墙直接登录数据库服务器。

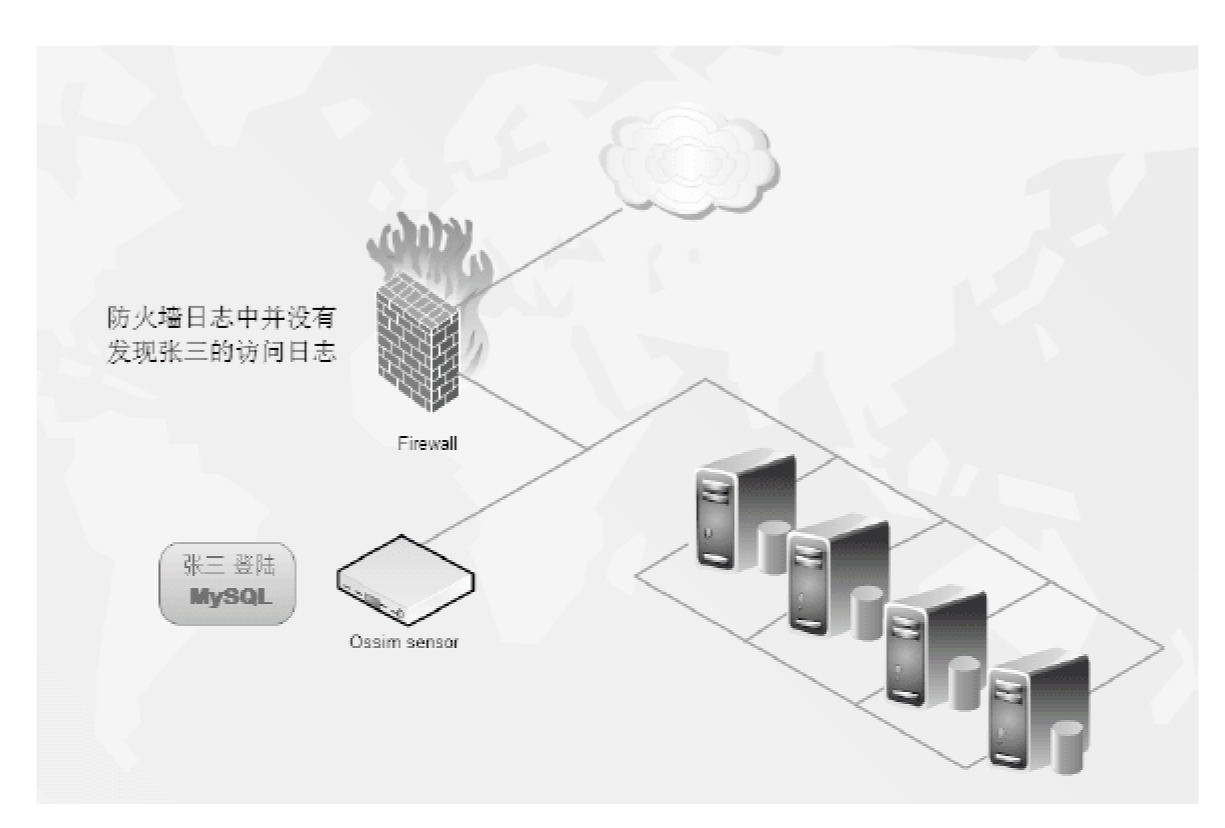


图 4-17 关联分析场景举例

（3）网络中 OpenVas 扫到某台 Linux 主机存在 Apache 2.2.x Scoreboard（本地安全限制绕过）漏洞，与此同时，NIDS 检测到了一个正在对该主机漏洞的尝试攻击事件，如果此时该 Linux 服务器打上了相应补丁，则关联分析结果为低风险值，并不会报警，如果没有打补丁，此时审计系统就会报警。

对于每套系统管理都有它的安全防护措施，只不过是安全孤岛，但是万物之间必然有它的联系，将这些日志联系到一起分析，就是我们上面讲的关联分析，这里关联的好坏就决定于它的关联库、关联规则和知识库。但是也有个矛盾，关联分析规则加载得越多，对系统影响越大，加载数量不够，也起不到作用。所以对于每个厂商而言，所使用的算法和关联分析引擎各不相同，没有统一的标准。

4.2.7 OSSIM 的关联引擎

为了达到安全事件关联分析的目的，就要有好的事件处理机制，比如前面讲的日志收集的

归一化处理，还得有好的关联方法，而且不止一种关联方法，将多种实时关联方法结合到一起效果更佳。大量标准化处理的事件被送入关联引擎处理后，它们会经历事件分类处理、聚合、交叉关联、启发式关联等多种关联方法，系统会根据数据库中的安全事件进行统计分类，找出经常导致安全事件的发源地和经常被攻击的端口，在这些阶段都会产生事件告警，其安全事件关联过程模块如图 4-18 所示。

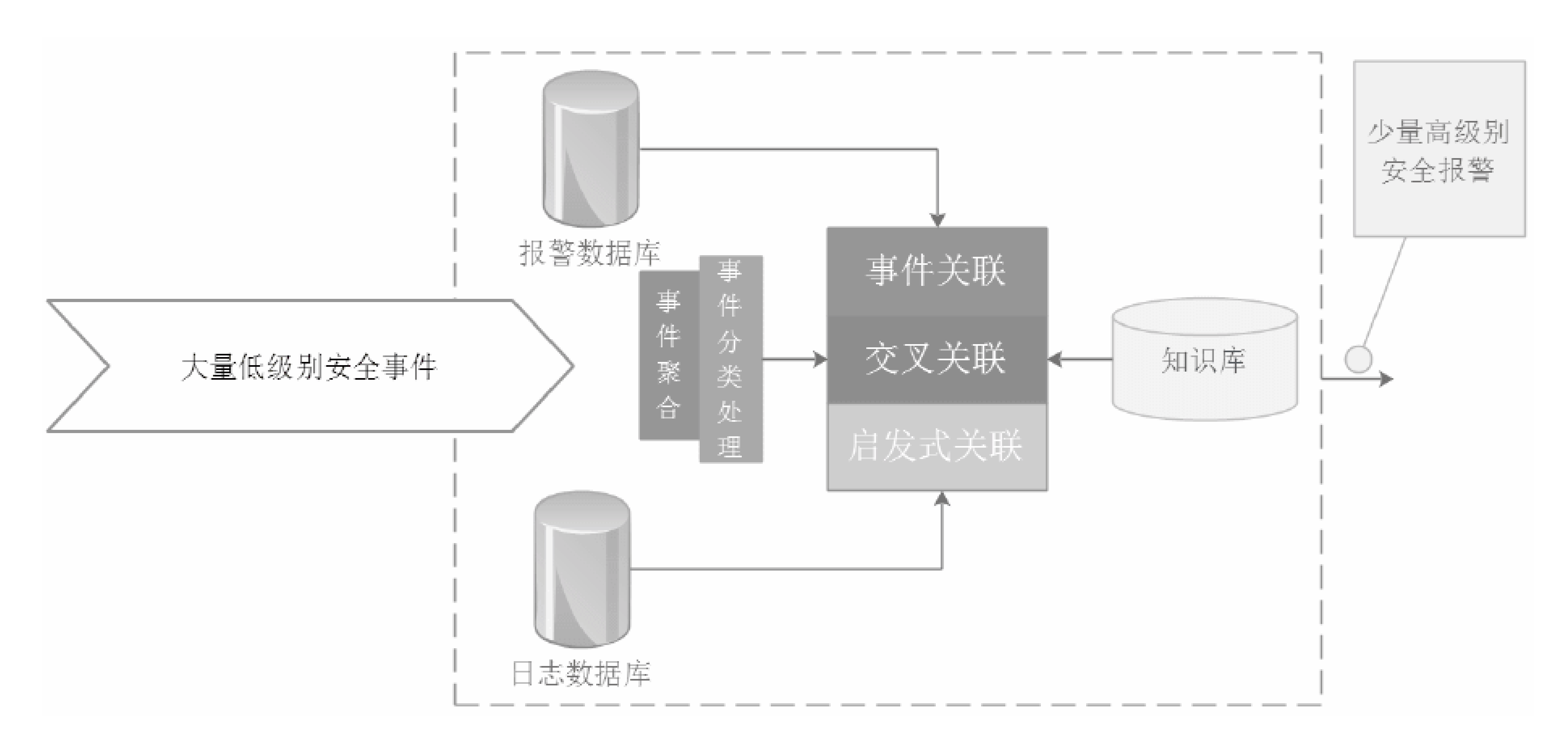


图 4-18　关联模块

聚合处理可以为后续关联提供高质量的安全事件，提高关联引擎效率。而分类处理可将已知可信度提高，受关注高的安全事件可直接提升为报警。而且分类通过对报警数据库、日志数据库、知识库综合进行分析统计，产生受控网段内最常被攻击的主机和端口分布，统计相同数据源和相同目标端口的安全事件的数量，这样就可以客观地反映网络异常情况。

在 OSSIM 系统中关联分析功能同样是由关联引擎来实现，关联引擎策略文件定义位于 /etc/ossim/server/，分析的数据由探针来收集。

探针（Sensor）每天要从网络上收集到成千上万的事件，如果对这些海量的事件信息不加任何处理就直接生成事件，这种做法是毫无意义的。而在报告之前通过关联分析可以将这些成千上万的事件进行浓缩（聚类），并确认成数十个甚至数个事件显示在 Web 前端的 SIEM 中。简单理解就是 OSSIM 的网络安全事件关联分析能将不同功能的开源网络安全检测工具产生的报警信息进行去伪存真，从而挖掘出真正的网络攻击事件。

定义关联策略的配置文件位于/etc/ossim/server/config.xml。

OSSIM 使用了两种关联引擎进行安全行为的关联分析，分别是基于事件序列的关联方法和启发式的关联方法。

一个基本的交叉关联规则举例：如果 Snort 发现基于一个目标 IP 的攻击，则说明该目标 IP 主机有漏洞，其可靠性系数为 10（即 100%攻击成功）。如图 4-19 所示中 REL 的值为 10，

即代表当前时间可靠性系数为 10，所以风险值为 4。这个值也可以在数据源中修改。对于启发式关联暂时放到 4.5 节再详细解读。

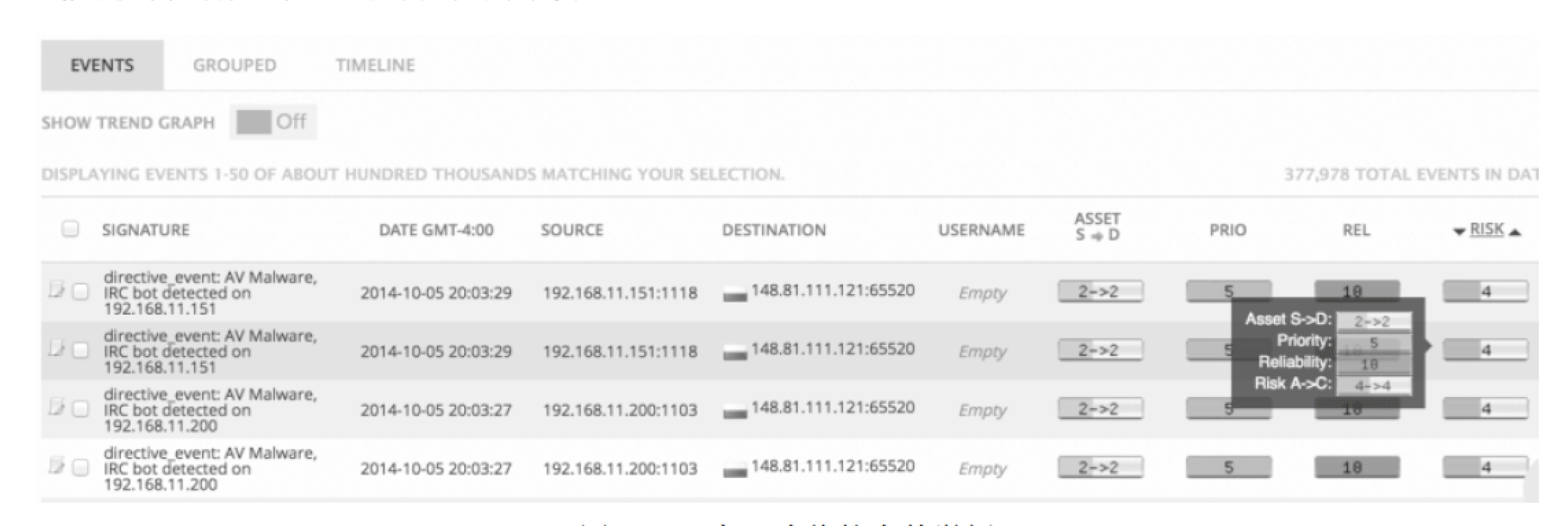


图 4-19　高风险值的事件举例

可靠性和优先级的数值决定了风险大小，手动修改数据源 REL/PRIO 的方法如图 4-20 所示。注意：这样修改后对全局有效，并不是仅对单条事件。

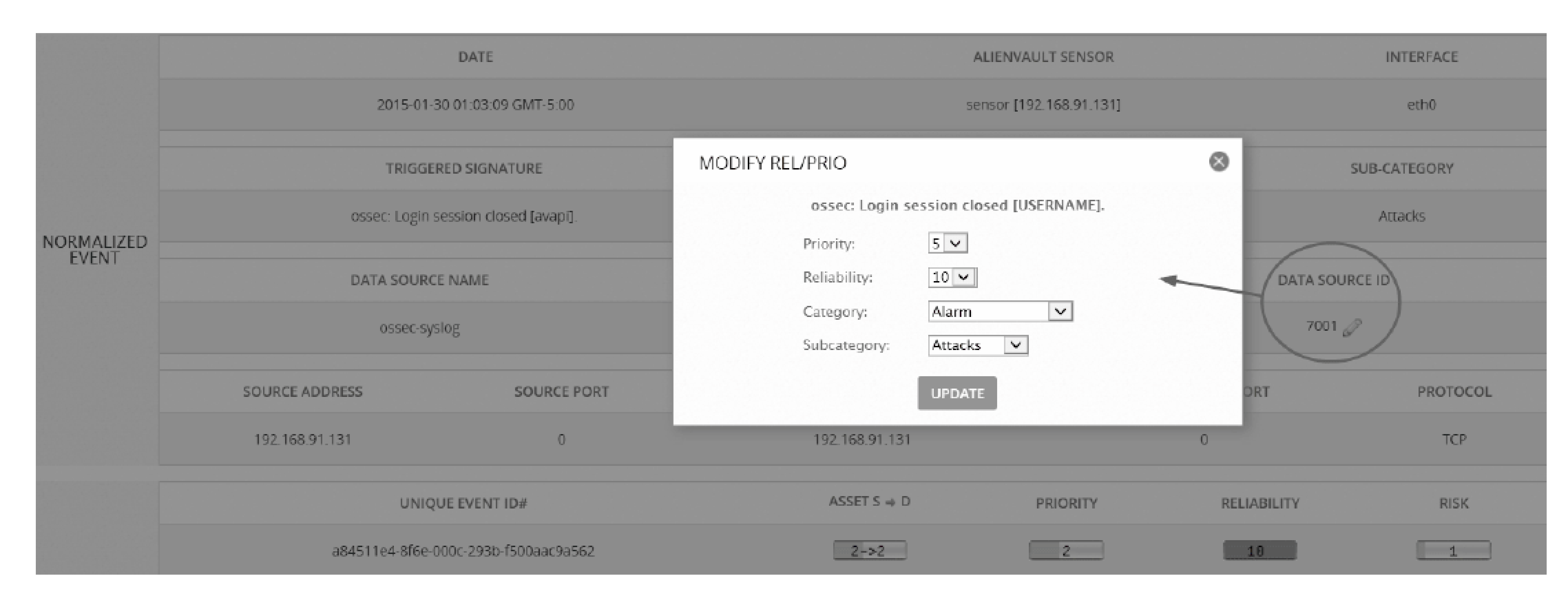


图 4-20　修改优先级和可靠性

4.2.8　事件的交叉关联

攻击总是作用于特定的网络环境，环境条件不具备时，攻击不能取得成功。对于那些即使是由真实攻击产生，但不具备攻击成功条件的安全事件，其威胁度也可置为很低。

本书第 6 章将分析，IDS 会产生大量误报的主要原因之一是，IDS 检测时基本不考虑现实网络的情况（如网络拓扑和主机信息等），那么单独使用 Snort 已经变得没有意义，在 OSSIM 平台中的做法是，通过 IDS 告警关联的数据模型，采用受监控信息系统的特性信息、漏洞信息、安全工具的信息以及安全事件四类信息进行交叉关联。交叉关联是确认告警非常有效的方法，当存在未知的漏洞以及漏洞扫描系统本身存在漏扫（没有发现系统存在的漏洞）和误扫（误认为系统存在漏洞）时，交叉关联会出现漏报和误报。

4.3 报警聚合

不同的报警有不同的报警格式，日志分析与解码是指针对各类日志（如系统日志与应用程序日志、防火墙日志、网络设备日志等）信息，按其各自的格式分别解析成统一的事件格式，从而为下一步的分析和处理做准备。下面是几种产品的报警日志信息样本。

4.3.1 报警样本举例

（1）Web Sever 的日志样本如下：

```
   ***.***.***.*** - - [03/Sep/2013:13:35:39 -0400] "GET
/scripts/..%c1%1c../winnt/
   system32/cmd.exe?/c+dir+c: HTTP/1.0" 404 315
   ***.***.***.*** - - [03/Sep/2013:13:35:39 -0400] "GET
/scripts/..%c0%af../winnt/
   system32/cmd.exe?/c+dir+c: HTTP/1.0" 404 315
```

（2）Snort 的报警日志样本如下：

```
   03/30-09:37:43.660306 [**] [1:1419:9] SNMP trap udp [**] [Classification:
   Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162 ->
   202.203.208.78:162
   03/30-09:37:43.940289 [**] [1:1852:3] WEB-MISC robots.txt access [**]
   [Classification: access to a potentially vulnerable web application] [Priority:
2] {TCP}
   68.142.250.14:37553 -> 202.203.208.104:80
   03/30-09:37:45.403954 [**] [1:1419:9] SNMP trap udp [**] [Classification:
   Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162 ->
   202.203.208.78:162
```

（3）Ossec 的报警日志样本如下：

```
   ** Alert 1226484882.185: - syslog,sshd,authentication_success,
   2013 Nov 12 18:14:42 lty-desktop->/var/log/auth.log
   Rule: 5715 (level 3) -> 'SSHD authentication success.'
   Src IP: 192.168.0.12
   User: lty
   Nov 12 18:14:40 lty-desktop sshd[5705]: Accepted publickey for lty from
192.168.0.12
   port 3255 ssh2
   ** Alert 1226486039.3272: - syslog,errors,
   2013 Nov 12 18:33:59 lty-desktop->/var/log/syslog
   Rule: 1006 (level 5) -> 'Syslogd restarted.'
   Src IP: (none)
```

```
User: (none)
Nov 12 18:32:57 lty-desktop syslogd1.5.0#1ubuntu1: restar
```

（4）防火墙的日志样本如下：

```
[08:11:37] 192.168.8.12 试图连接本机的 3498 端口
TCP 标志：S，该操作被拒绝。
[12:55:07]接收到 192.168.8.15 的 UDP 包，本机端口：1214
对方端口：OICQ S 的[8000]该包被拦截。
[13:12:02]192.168.8.12 尝试用 Ping 来探测主机，该操作被拒绝。
```

由上面的报警日志信息，我们可以看出各种日志相互独立，它们之间缺少统一的互动合作，所以各自产生出来的报警日志格式及信息会有所不同，这就不利于我们对整个网络安全事件进行查看和掌握，肯定不乏大量的冗余信息，因此不利于我们对报警事件集进行聚类与关联分析，攻击场景的构建及攻击意图的挖掘，严重影响了系统的检测结果和实时监控性能。

所以我们需要对这些报警信息进行日志的分析与解码，去除一些正常的访问记录和误报信息，提取出真正报警信息里的关键字段，并将之存储为统一格式，便于下一阶段聚类与关联分析。

4.3.2 事件聚合

不同的攻击行为，产生不同的报警信息，由于安全产品各式各样，报警的格式会有所差异。下面我们用不同的方法对安全产品产生出来的报警信息进行聚类与合并。在实际应用中，采用基于主机和基于网络的 IDS 混合检测方式，对主机和网络进行监测。基于主机的监测软件用 Ossec，基于网络的借助于 Snort 监控软件来完成。主机监控软件 Ossec 可以完成主机上的审计记录、系统日志、防火墙日志、应用程序日志、Rootkit 检测等分析，伴随着对整个主机的监控日志 ossec-alert.log。

网络入侵检测软件 Snort 负责对网络上的数据包进行深度检测，它由三个重要的子模块构成：

- 数据包解码器模块
- 检测引擎模块
- 日志与报警系统模块

Snort 主要工作模式有三种，分别是：嗅探器模式，数据包记录器模式和网络入侵检测系统模式。嗅探器模式仅从网络上读取数据包，并作为数据流显示在终端。

数据包记录器模式的主要工作是把在网络上收集到的数据包记录到硬盘。另外，Snort 的配置是可以按照用户的需求，根据自己网络环境的具体情况和不同而设定自己想要的功能，配置比较灵活。因此，可以通过让 Snort 分析网络数据包来匹配用户自定义规则，并根据检测结果采取行动，详情参见第 6 章。

4.3.3　事件聚合举例

我们对上述两类检测工具的报警信息进行分析后，看出 Ossec 报警信息主要由 IP 地址、登录的用户和对此主机的操作内容的记录、报警级别、报警时间、用户名和规则 ID 等几部分组成。根据报警信息所包含的这些特征属性，我们可以指定一系列规则，对这些报警信息进行解码，再进行模式匹配，从而把报警事件进行分类，对主机入侵检测软件 Ossec 的报警信息，我们可以通过其规则 ID 找到入口，以报警类别进行分类。

比如下面这条 Ossec 的报警日志，表明日志服务器重新启动。触发的规则编号为 1006，报警时间是 2013 年 9 月 12 日下午 18 点 33 分 59 秒，报警级别为 5 级，报警的大类别属于 syslog，子类为 error。

```
** Alert 1226486039.3272: - syslog,errors,
2013 Nov 12 18:33:59 lty-desktop->/var/log/syslog
Rule: 1006 (level 5) -> 'Syslogd restarted.'
Src IP: (none)
User: (none)
Nov 12 18:32:57 lty-desktop syslogd1.5.0#1ubuntu1: restart.
```

负责对数据包进行监控的 Snort 报警日志主要由时间、IP 地址、通信协议、事件 ID、报警级别和报警事件描述等几部分组成。由于同一种攻击，一般只采用一种网络协议的方式，所以我们可以对网络入侵检测工具 Snort 的报警信息采用基于协议的方式进行分类。例如从下面的几条报警中，我们可以看到主机正在遭受网络探测工具的攻击，利用的协议是 ICMP 协议。

```
07/16-11:58:41.160021 [**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 221.217.44.192 ->
202.203.208.104
07/16-12:04:26.068608 [**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 207.190.60.251 ->
202.203.208.104
07/16-12:14:27.113751 [**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 218.255.241.205 ->
202.203.208.104
```

再如下面这几条攻击行为，通过其报警信息看出来，网络系统正在遭受着针对 CGI 漏洞的扫描攻击等，其中所利用的通信协议是 TCP 协议。

```
03/29-18:50:22.491474 [**] [1:1113:5] WEB-MISC http directory traversal [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
220.164.125.98:49245 -> 202.203.208.104:80
03/29-18:50:25.609606 [**] [1:1113:5] WEB-MISC http directory traversal [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
220.164.125.98:49314 -> 202.203.208.104:80
03/29-18:50:25.719474 [**] [1:1113:5] WEB-MISC http directory traversal [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
220.164.125.98:49316 -> 202.203.208.104:80
```

```
    03/29-18:50:25.745592 [**] [1:1113:5] WEB-MISC http directory
```

下面的几条 Snort 的报警是基于 UDP 协议的、针对 CGI 漏洞的攻击。

```
    06/02-19:34:26.740820 [**] [1:1419:9] SNMP trap udp [**] [Classification:
    Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162
->202.203.208.78:162
    06/02-19:34:26.740906 [**] [1:1419:9] SNMP trap udp [**] [Classification:
    Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162
->202.203.208.78:162
    06/02-19:34:27.083436 [**] [1:1419:9] SNMP trap udp [**] [Classification:
    Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162
->202.203.208.78:162
    06/02-19:34:27.083450 [**] [1:1419:9] SNMP trap udp [**] [Classification:
    Attempted Information Leak] [Priority: 2] {UDP} 202.203.131.179:162
->202.203.208.78:162
```

总之，针对不同的报警信息应该采用不同的聚类与合并的方法，对冗余报警信息进行处理。聚类的依据可以是报警日志中的关键字、事件 ID 和攻击特征等。分别识别出各类攻击，如 CGI 攻击、缓冲区溢出、DOS 攻击和 SQL 注入攻击等类。

4.3.4 事件聚合在 OSSIM 中的表现形式

网络安全事件的关联分析技术是对不同地点、不同时间、不同设备上的网络告警进行多维度的分析，对于事件聚合的目标就是消除重复报警。下面我们观察 OSSIM 如何采用图形化方式展示出来。首先查看 Alarm 分组效果，如图 4-21 所示，SIEM 分组如图 4-22 所示。

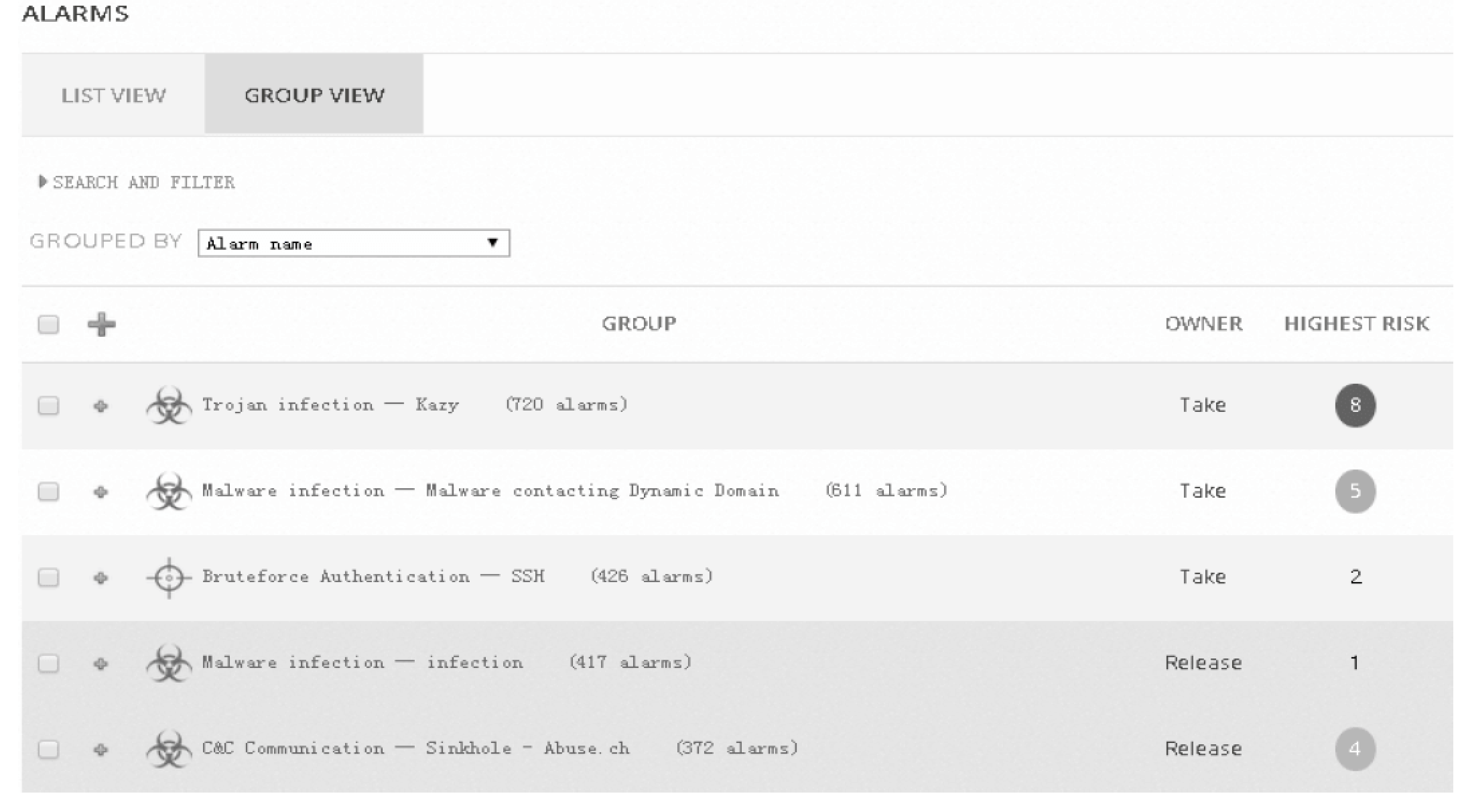


图 4-21 Alarm 分组聚合

DISPLAYING EVENTS 1-50 OF ABOUT HUNDREDS MATCHING YOUR SELECTION.

SIGNATURE	TOTAL # (*)	UNIQUE SRC. #	UNIQUE DST. #	LATEST EVENT
SSHd: Received disconnect	75936	-	-	
SSHd: Login sucessful, Accepted publickey	75936	-	-	
ASA: The specified host tried to access the specified URL	5691	-	-	
ASA: An address translation slot was created	4455	-	-	
Deny inbound (No xlate) string	3825	-	-	
ASA: A TCP connection slot between two hosts was created	3190	-	-	

图 4-22　SIEM 分组聚合

图 4-21、图 4-22 所给出的仅是一些典型攻击报警案例，在实践中黑客攻击步骤复杂，我们通过构造好的关联规则，才能将多条告警还原成一个完整的攻击场景，问题来了，到底如何构造关联规则呢？参见 4.7 节内容。

4.3.5　SIEM 中的冗余报警

冗余报警的产生途径分两种，一种是由于多个安全产品的同时使用，所以会出现同一个攻击被多个检测引擎同时检测出来的现象，另一种是相同攻击被一个安全产品多次检测出来。

合并冗余的报警信息主要从 3 方面来考虑：

（1）合并基于主机的监控 Ossec 产生的冗余报警事件。

（2）合并基于网络的监控 Snort 产生的冗余报警事件。

（3）合并主机监控 Ossec 和网络监控 Snort 对相同攻击产生的报警信息。

Snort 报警信息比较全面，特征明显，对 Snort 产生出来的报警事件合并的过程可以采用基于属性相似度的方法。而对于 Ossec 的报警事件的合并，OSSIM 采用基于事件 ID 和报警类别信息相结合的方法。目前 Ossec 有 900 多条检测规则，均以 XML 格式存储。经过统计分析之后，可以按报警的行为来分类，共分为 80 个报警大类，常见的包括 Syslog、Firewall、IDS、Web、Squid、Windows 等。以 OSSEC 报警的事件 ID 为入口，从而进行逐级的类与子类之间的匹配，以达到匹配与合并的目的。

4.3.6　合并相似事件

聚合的对象是基于 Snort 和 Ossec 所产生的报警，由于 Snort 的报警数据（包括 IP 地址，攻击内容和端口等）中含有协议属性，可采用基于规则的方式聚合，在聚类过程中可以保证报警信息得到正确的归类。

对于 Ossec 产生的报警信息，其报警事件的 ID 为入口，逐步按报警类别进行聚合，合并出来的误差低。以 ID 为入口按根类别进行合并的方式，节省了遍历报警信息进行聚类与合并的时间。

OSSIM 采用了 Ossec 来实现对主机的审计记录，系统日志和应用程序的采集分析。同时

使用 Snort 对网络上的数据包进行采集，完成实时流量分析和对网络上的 IP 包进行测试等功能，也可进行协议分析、内容查找和匹配，能用来探测多种攻击和嗅探（如缓冲区溢出和 ShellCode 攻击等）。

4.3.7 同类事件的判别

对于两条事件而言，如果它们的相似程度较高，那么就认为它们是同一个攻击而进行合并，否则就认为是互不相关的攻击行为。这种相同的相似度报警如图 4-23 所示，不同的相似度报警如图 4-24 所示。

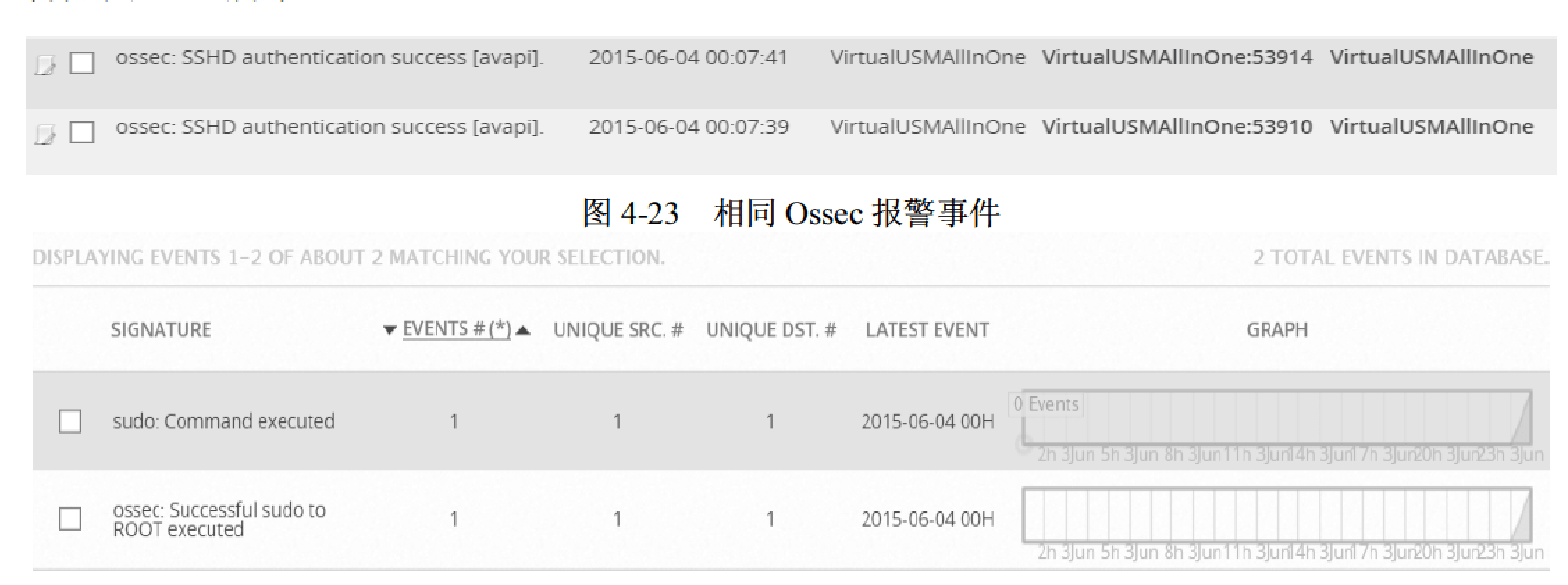


图 4-23 相同 Ossec 报警事件

图 4-24 不同 Ossec 报警事件

该思路如何让计算机识别呢？可以从计算这两条报警信息之间的相似度入手，首先考虑这两条报警的共有属性。这些属性包括攻击源、目标（主机 IP 和端口）、攻击类型、MAC 地址、时间信息和事件 ID 等。据此我们可以对报警事件的属性进行抽象化建模，这种模型的流程如图 4-25 所示，将它们的属性定义为一个事件属性集。我们对冗余的报警事件所采用的合并方法的出发点就是，对相同的攻击行为的报警信息之间存在着很高的相似度，对不同的攻击行为的报警相似度较低。

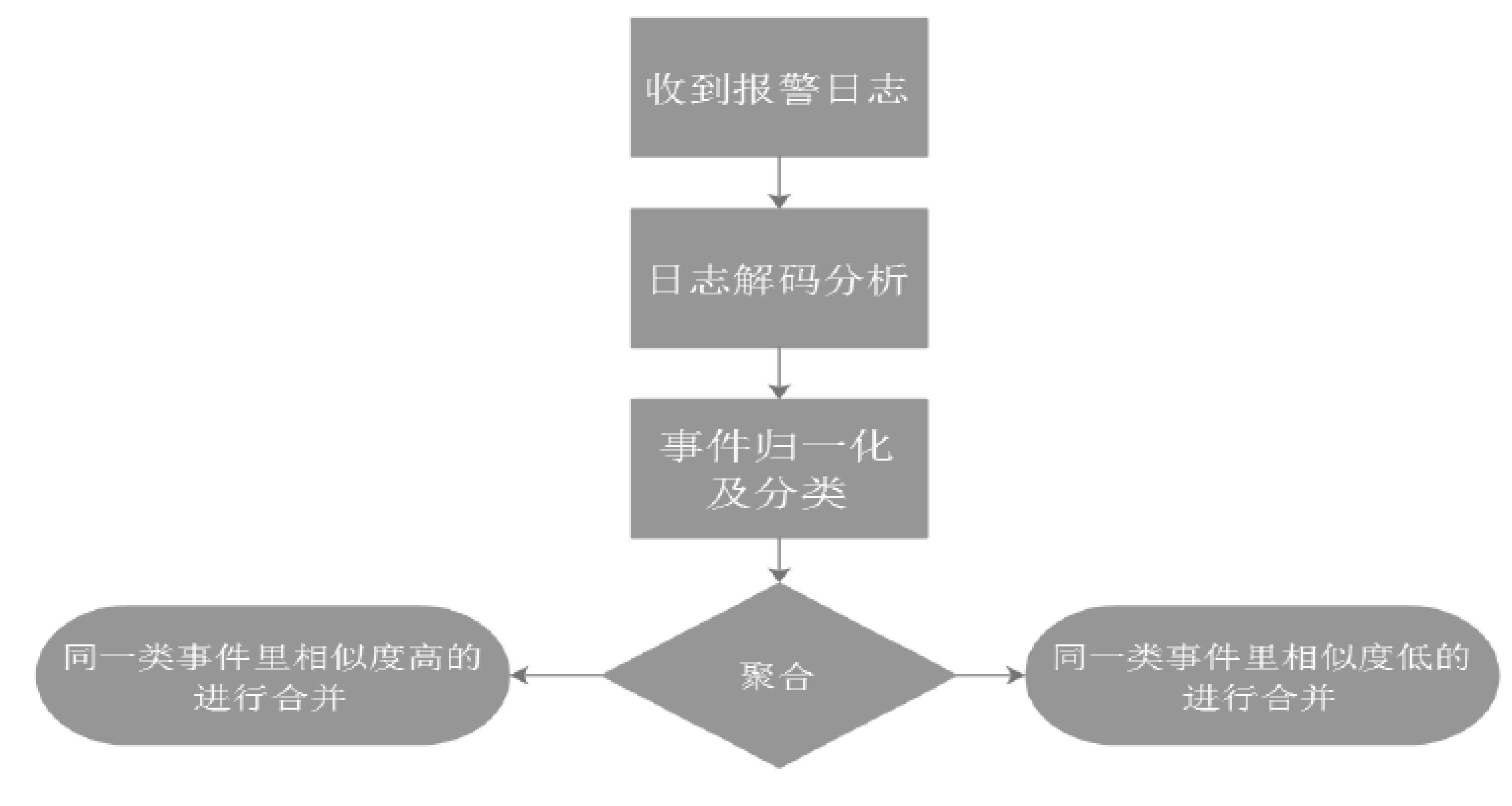


图 4-25 聚合流程图

4.3.8 合并流程

以 Ossec 事件的聚合为例，其合并的算法步骤可以描述为：

（1）根据已知知识定义报警数据库；

（2）接收新的报警事件并进行分析与解码；

（3）对新报警事件解码并按照协议号或事件 ID 和内容进行格式的规范化；

（4）在 IDS 里相同的攻击行为总是在一段连续的时间内发生的，时间越接近相似程度会越高。另外，考虑到在计算属性的相似度时，要用到时间属性，所以将新加入的报警（第 n+1 条）在同一类中逐个与旧报警信息（共 n 条）计算属性的相似度。

在计算时采用比较法，即先计算新信息和第 n 条报警事件的属性相似度，如果不满足合并，接着和第 n-1 条做比较，依次迭代到第 1 条。

4.3.9 事件映射

实践中发现对一些攻击行为，Snort 和 Ossec 会产生报警现象，比如 Synflood 攻击。但对于其他一些攻击行为，Snort 和 Ossec 会同时产生报警，比如 Nmap 扫描。对于由 Ossec 和 Snort 工具所产生的、对同一攻击行为的重复报警信息的合并，采用映射规则的方法。其映射原理如下：

（1）自定义一个报警规则集，称为内部规则集，赋予每条内部规则集一个编号及对应的描述。

（2）由于 Ossec 的规则有 900 多条，所以可以把 Ossec 的每一条报警规则都用基于规则 ID 的方式映射到内部规则集中，即把 Ossec 的报警规则和内部自定义的规则逐条对应。

（3）Snort 的报警规则有 9000 条，逐条地实现映射到内部规则集的任务量太大，可按报警的类别和基于服务分为 30 个大类报警规则，其中每个大类里面又含有小类。

（4）将 Snort 的报警规则按报警类别和基于的服务也逐条地映射到内部规则集中。

这样通过上面过程从而使重复的报警信息量进一步降低。

4.3.10 Ossec 的报警信息的聚类

从 Ossec 事件 ID 入口，根据报警类别进行合并，其中一些报警信息只有一个根类别，比较容易。

```
   ** Alert 1326486837.2066: - ossec,2014 Nov 12 10:30:57
lty-desktop->ossec-logcollector
   Rule: 591 (level 3) -> 'Log file rotated.'
   Src IP: (none)
   User: (none)
   ossec: File rotated (inode changed): '/var/log/syslog'.
```

有些报警信息类别间的层次关系不是一层，例如，下面这条针对缓冲区管理漏洞的攻击的一条 Ossec 报警信息。

```
   ** Alert 1226484882.185: - syslog,sshd,authentication_success,2014 Nov 12
10:14:42 lty-desktop->/var/log/auth.log
   Rule: 5715 (level 3) -> 'SSHD authentication success.'
   Src IP: 192.168.0.120
   User: lty
   Nov 12 10:14:40 lty-desktop sshd[5605]: Accepted publickey for lty from
192.168.0.120 port 3256 ssh2
```

从报警信息中，我们可以看到，报警信息的类别层次关系分为三层：syslog、sshd、authentication_success。其中 Syslog 日志服务器做为根类别，sshd 进程做为子类别，authentication_success 信息是子类别中的下一级子组。这类的信息聚类起来就要求做对比的次数要多，聚类查找时的深度要比第一类报警更深，耗费时间更长。

在上面的报警信息中，我们可以看到报警时间字段：2014 Nov 12 10:14:42 和报警级别：level 3 等一些信息，通过这些信息之间的关联对事件进行合并操作，其合并流程如图 4-26 所示。

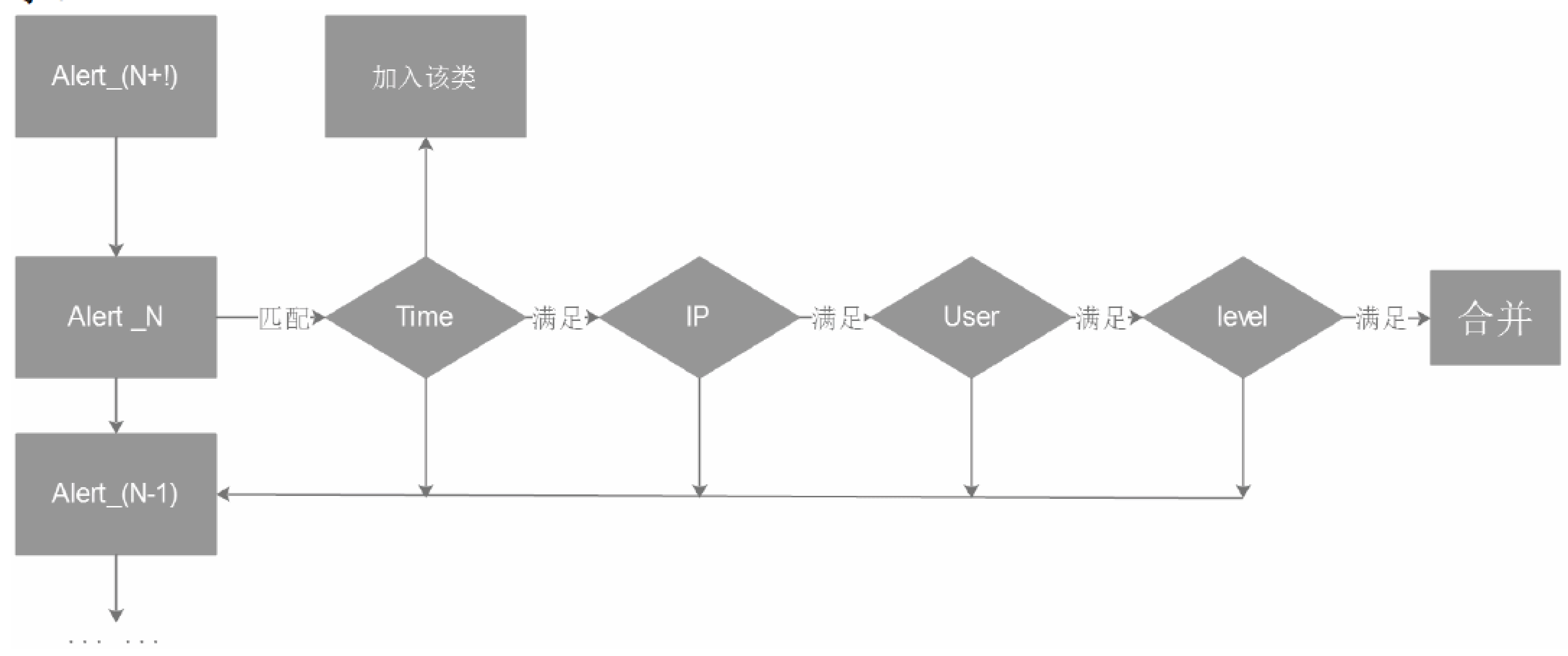


图 4-26　合并流程图

合并 Ossec 的重复信息实现过程在其子类中进行，对字段 user、ip 和 level 采取逐个校验方式，这 3 个字段只要有差异，新接收的报警就会转入下一节点的验证合并操作。对于攻击发生的时间字段，我们设定一个范围，如果这两个报警事件的时间差的绝对值超出了所设定的范围，可认定这两个报警事件是两次攻击中的事件，不是同一次攻击中的重复信息，但新的攻击报警事件的可信度会提高。如果遍历子类中的事件无法合并，则把此信息加入该子类作为该子类的新元素。

4.3.11　Ossec 与 Snort 事件合并

由于 Ossec 对网络流量不敏感，而 Snort 对网络上的数据包起到检测作用，所以对于大部分的简单攻击行为来说，Ossec 和 Snort 只会有一个程序对攻击进行报警。随着攻击方式的改变，现在很多攻击行为融合在一起，采取复杂的攻击方式进行入侵。

所以对于一些复杂的攻击事件，Ossec 和 Snort 会同时对一个攻击行为产生报警。对于二者的合并，我们采用借助中间人的方式来完成，即定义一个内部的报警行为规则表。

通常攻击的过程分为 3 个阶段，即预攻击阶段（端口扫描，漏洞扫描和系统鉴别等信息收集过程），攻击实施阶段（访问注入，提升权限和拒绝服务等），攻击后阶段（窃取数据，安装后门程序和篡改文件等）。

其中定义了内部规则表，对 Ossec 的 900 多条规则逐个进行映射，把这些规则根据报警事件中攻击行为的目的，分别映射到内部规则上，读者可以打开/etc/ossim/agent/plugins/ossec-single-line.cfg 文件，找到[translation]项，其中包括 31151=7014、5706=7014、31163=7014。

上例中的第一条映射规则就是把 Ossec 的事件 ID 号为 31151 的报警规则映射到 7014。

4.4 风险评估方法

4.4.1　风险评估三要素

风险评估（Risk Assessment）对网络内的资产及整个网络进行的风险评估。风险评估是包括威胁、资产和漏洞的评估，其目的是帮助识别资产，分析判断其价值、存在的脆弱性和面临的安全威胁，提高安全体系的投资效率。

OSSIM 系统中的风险评估主要围绕着威胁（Threat）、漏洞（Vulnerability）、响应（Response，即安全措施）、资产（Asset）来监控。风险评估模型嵌入在 OSSIM 关联引擎系统中，关联引擎结合知识库中的资产库、漏洞库、威胁库将风险的资产、漏洞、威胁三个要素用维度表现出来，它们的关系如图 4-27 所示。

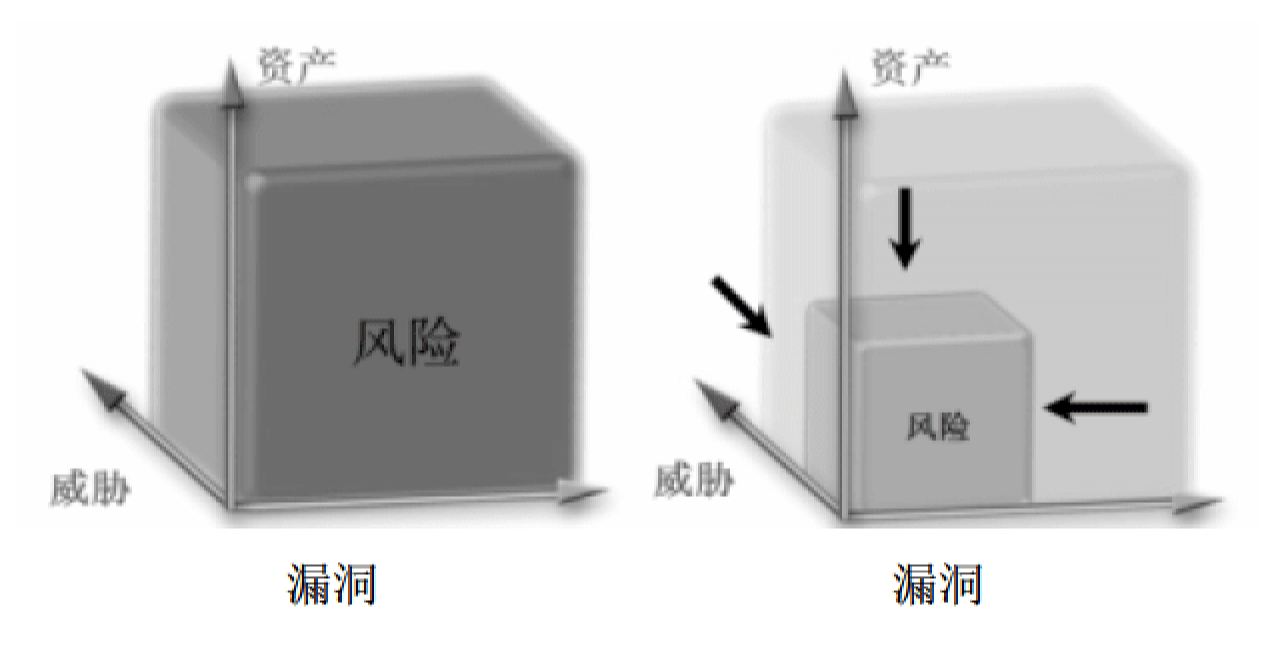


图 4-27　风险的三维度

（1）资产（Asset），它是需要保护的对象。

- 网络设备：路由器、交换机等。
- 终端设备：服务器、工作站等。
- 应用系统。
- 数据。

（2）漏洞（Vulnerability），它是物理布局、组织、人员、管理、硬件、软件或信息中存在的缺陷与不足，它们不直接对资产造成危害，但脆弱性可能被环境中的威胁所利用从而危害资产的安全。

（3）威胁（Threat），威胁会对资产造成危害，可能是人为的或非人为的，可能是无意失误，也可能是恶意攻击。

降低风险就需要挤压风险（A、V、T）三个值，也就是需要从三个维度共同发力。

```
Risk=R(asset,vulnerability,threat)
```

将资产价值（Asset）、优先级（Priority）、可靠性（Reliability）三个参数组合进行风险的计算是简洁有效的，在 OSSIM 系统中使用以下公式：

```
Risk=Asset*Priority*Reliability/25 （下文称风险模型计算公式4-1）
```

- Asset（资产，取值范围 0~5）
- Priority（优先级，取值范围 0~5）
- Reliability（可信度，取值范围 0~10）

由上面风险模型计算公式（4-1）计算每个 Alert 事件的 Risk 值，其中：

- Asset 的取值范围为 0～5，Asset 默认值为 2。在 OSSIM 系统中将资产的关注程度分为 5 级，取值由低到高分别为 1、2、3、4、5。从表面上理解，数字的大小决定了风险计算公式中 Risk 值的大小，但也有更深层的含义。有很多企业并不知道其关键资产是什么，也不知道如何保护。比如普通工作站资产等级为 1，当它遭受 DOS 攻击时，我们可切断网络连接。如果是数据库服务器，它的资产等级为 5，数据库服务需要实时在线，所以同样遭受 DOS 攻击，我们就不能像工作站那样处理，而应自动启用备用 IP 地址，并将攻击引向网络蜜罐系统进行分析。
- 优先级 Priority 的取值范围为 0～5，默认值为 1。该参数描述一次成功攻击，所造成的危害程度，数值越大，则危害程度越高；
- 可信度或者叫可靠性 Reliability，取值范围为 0～10，默认值为 1。可靠性参数描述一次攻击成功的概率，最高值是 10，代表 100%可能，所以其值越高，代表越不可靠，大家也可以将此理解为被攻击成功的可能性。

大家在操作 OSSIM 时，打开 Web UI 中的 SIEM 控制台，观察每条 Event 时，即能发现风险值由资产值优先级和可信度来控制。网络中每个资产都是有价值的，这个价值的量化就通过

资产值来实现，每个默认资产为 2（范围 1~5），从风险计算公式中可以分析出，风险计算不将资产值作为影响风险结果的主要因素，例如某些数据库服务器资产值很高，则计算出的风险值也会很大。而那些资产值小的工作站，在遭受严重攻击的节点风险值小，这样很难被关注，从而失去原本的真实性，所以让资产值的范围在 1~5 之间，影响风险值也就小。

在系统中会计算出两个风险值 Risk_A、Risk_C，如果资产价值是源主机的 asset 值，那么计算结果就是 Risk_C，如果是目标主机的 asset 值，那么就是 Risk_A。有关这两个值在 alienvault 库 event 表中的 risk_a、risk_b 字段中。如果对具体算法感兴趣，请阅读 /usr/share/ossim/include/classes/asset.inc 文件。

例如，一台主机在 VLAN 链接了 5 个不同主机的 445 端口，这可能是正常通信，如果连接了 15 台机器的 445 端口，这就比较可疑了，如果有几百个这样的连接并且持续时间很长，那就基本可以确定是受到蠕虫攻击。

4.4.2　Risk & Priority & Reliability 的关系实例

在我们收集到的事件中，多数情况下低风险级别的事件占多数，比如普通的扫描事件，以 Cisco ASA 的一个 ICMP 事件为例，其 PRIO=1，REL=1，ASET=2，根据风险模型计算公式 4-1 计算结果近似为 0。这种扫描事件特征和原始日志截图如图 4-28 所示。

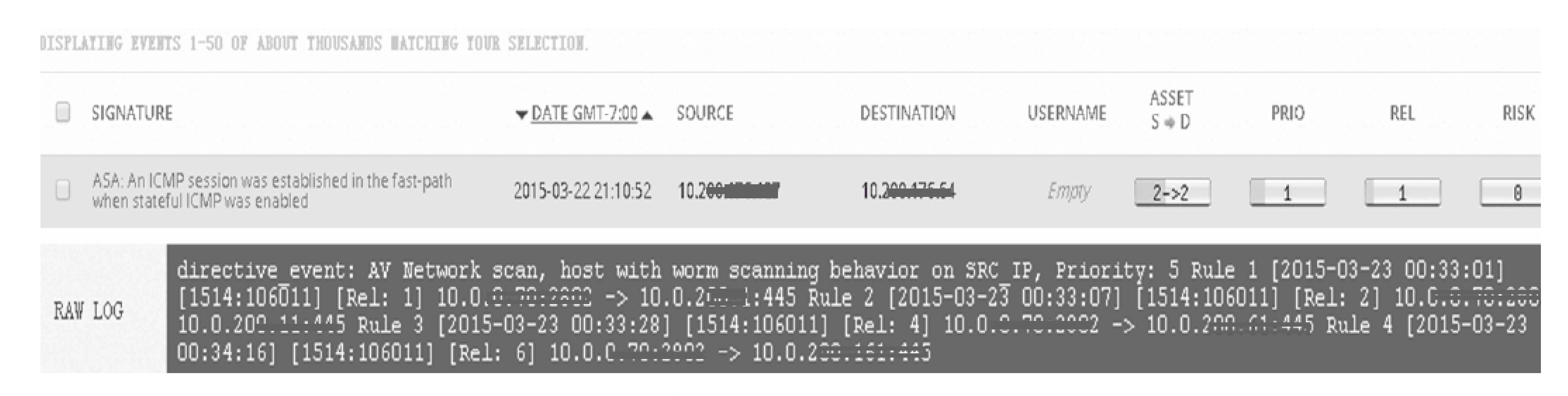


图 4-28　同时显示一条事件的风险值、优先级与可靠性数值

这种低风险事件并不会引发报警，如果是 Windows 下的蠕虫对 445 端口的扫描结果就大不相同，风险级别上升为 3，如图 4-29 所示。

EVENT DETAILS

NORMALIZED EVENT

DATE	ALIENVAULT SENSOR	INTERFACE
2015-03-22 17:34:16 GMT-7:00	Unknown	-

TRIGGERED SIGNATURE	EVENT TYPE ID	CATEGORY	SUB-CATEGORY
directive_event: AV Network scan, host with worm scanning behavior on 10.0.0.70	30045	Alarm	Scan

DATA SOURCE NAME	PRODUCT TYPE	DATA SOURCE ID
directive_alert	Alarm	1505

SOURCE ADDRESS	SOURCE PORT	DESTINATION ADDRESS	DESTINATION PORT	PROTOCOL
windows70	2802	10.0.200.161	445	TCP

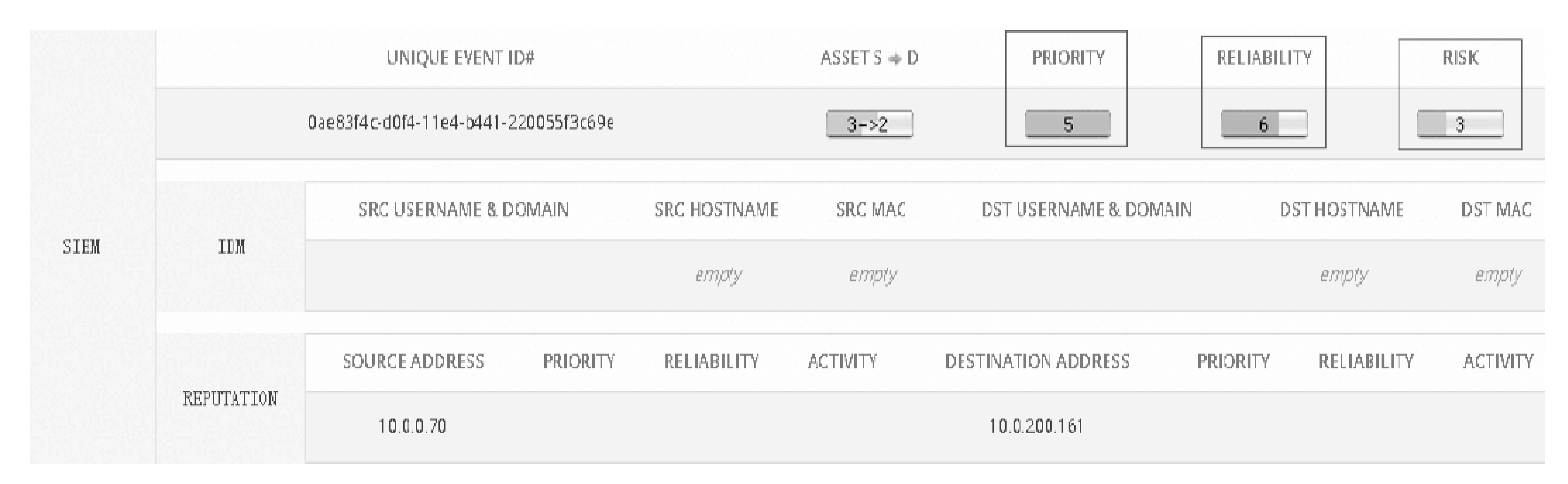


图 4-29　一条事件对应的详细优先级与可靠度的数值

更高风险的事件分析，例如木马成功攻击了某系统，这时风险级别达到 5 以上，图 4-30 中显示的 Risk 值为 8，系统用红色醒目字体标出。

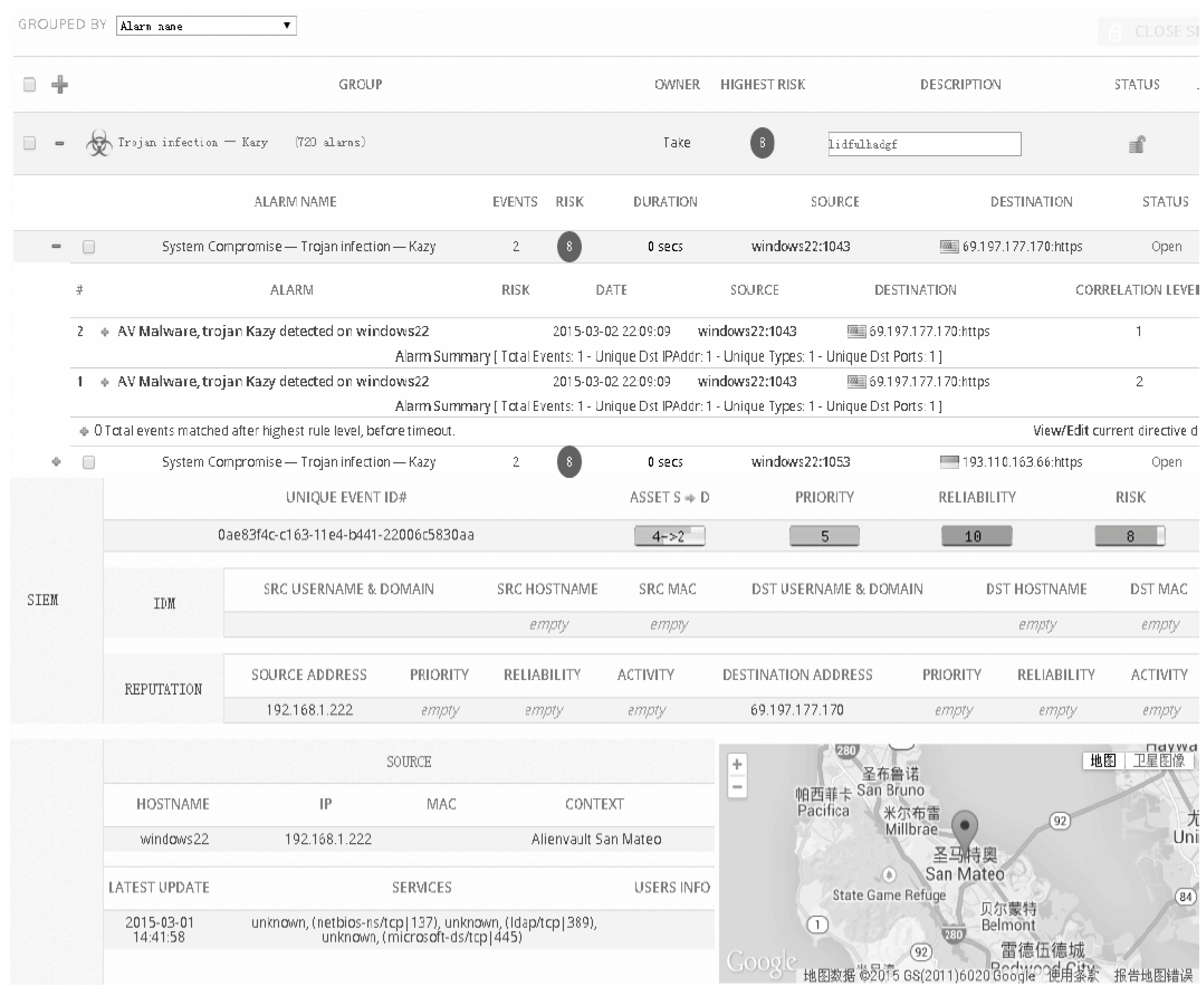


图 4-30　Risk 为 8 的报警事件

在 OSSIM 平台中对资产的哪些特征进行监控？大家可在 OSSIM Web UI 下 Environment→Assets 进行查看，这些元素可以归纳为如图 4-31 所示，由 HIDS、Plugins、Netflow、Service 等元素统统与资产相关联，这与黑客攻击成功与否有着直接联系。

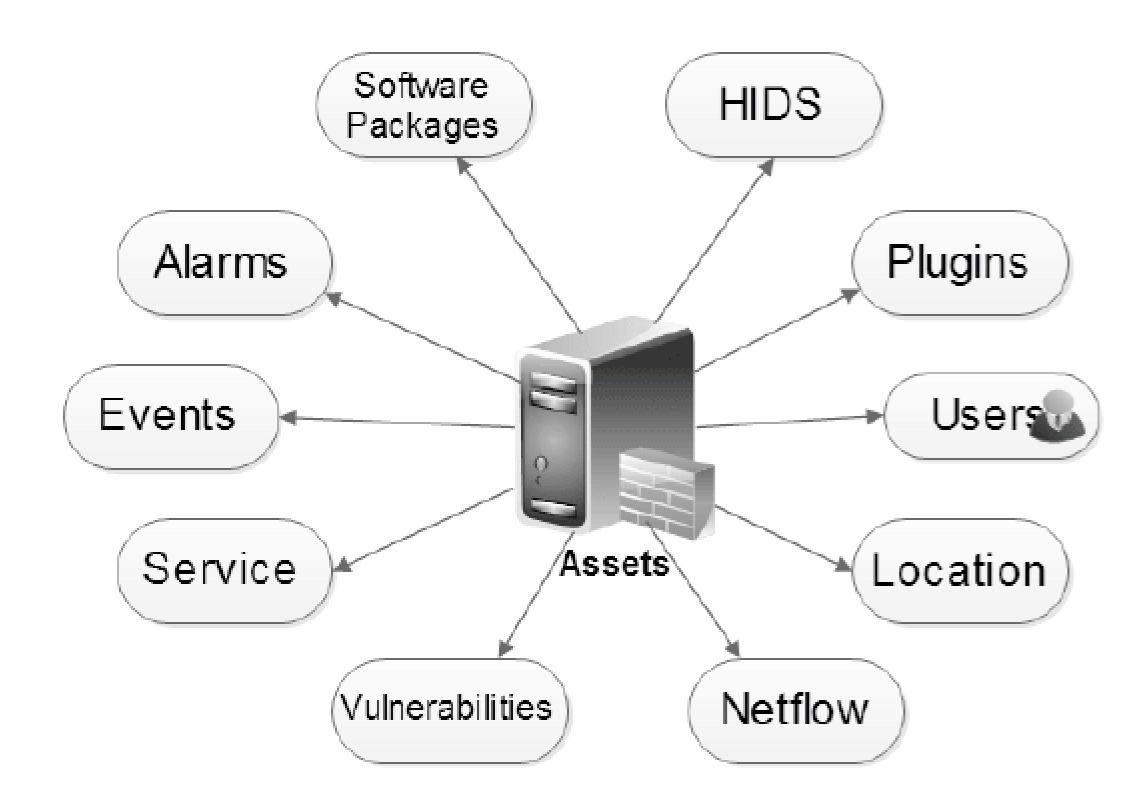


图 4-31　资产关联特征

一次成功的攻击，需要有如下几个因素：

- 目标主机有该攻击成功所需要的漏洞。
- 操作系统匹配。
- 受攻击的服务在运行。
- 端口打开。
- 应用程序在运行。

以上这 5 个因素构成了攻击图，如图 4-32 所示。

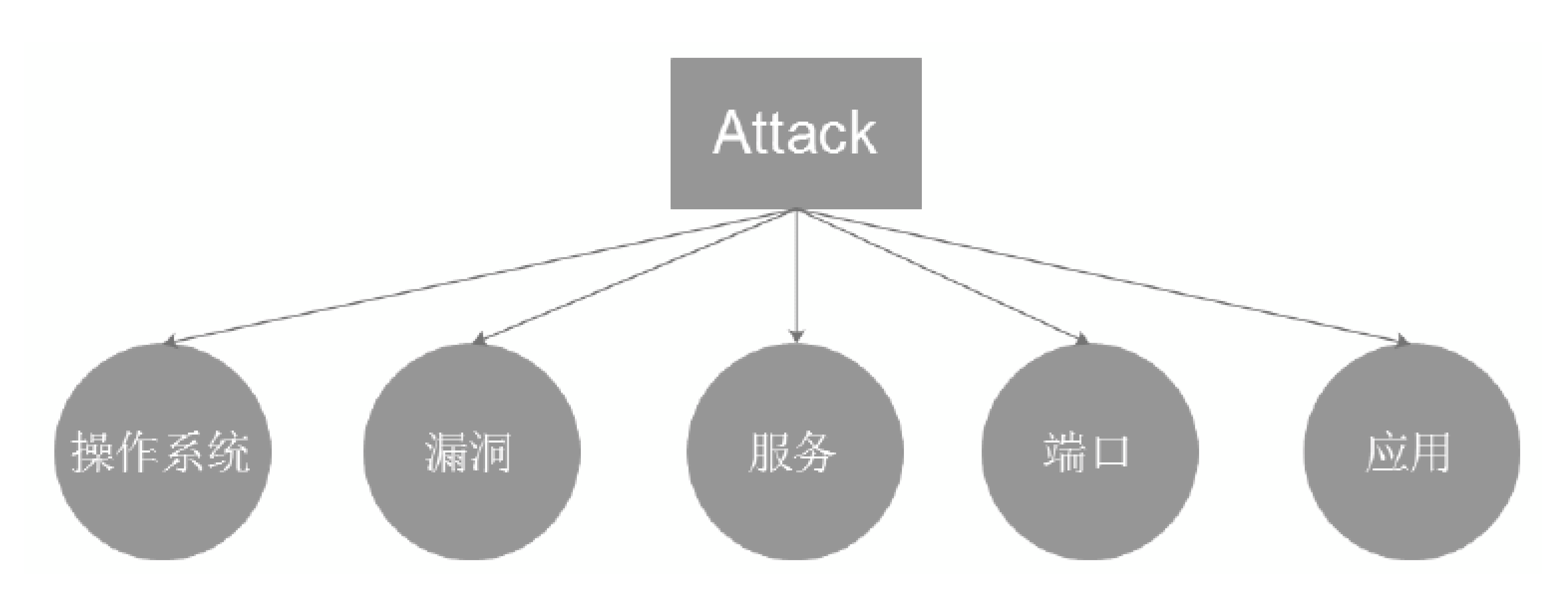


图 4-32　构成攻击的 5 要素

很多企业并没有花时间去将他们的内部资产进行整合，更没有关联分析，他们往往关心某一个方面的指标，例如网卡的流量，所以他们无法有效地确定可疑的内部威胁，以及周边环境的安全性。

有些日志消息本身具有优先级，而有些则可能没有，原则上是给最紧急事件最高的优先级，因为有些低级别的日志是不需要关注，而高优先级日志才需要关注，如果每个日志消息都包含优先级信息，那么开销将会很大，下面看个简单的例子。

Cisco ASA with FirePower Services（思科收购 Sourcefire 的新产品）设备产生的一条日志如下：

```
alienvault:/var/log# tail syslog
Dec 28 19:46:19 IDS01 SFIMS: [CA IDS][Policy1][119:15:1] http_inspect: OVERSIZE
```

```
REQUEST-URI DIRECTORY [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
10.12.253.4:55504 -> 94.15.224.60:80
    Dec 28 19:46:23 IDS01 SFIMS: [CA IDS][Policy1][128:4:1] ssh: Protocol mismatch
[Classification: Detection of a Non-Standard Protocol or Event] [Priority: 2] {TCP}
56.16.69.166:59578 -> 10.87.156.248:22
```

4.4.3　动态可信度值（Reliability）

在关联规则中 Reliability（可信度）通常有个初始值，试想如果 Reliability 是个常量，那么在动态的网络攻击中将会产生大量误报，在引入交叉关联机制之后，就将网络安全报警和具体网络应用服务、开放端口、开放端口的漏洞信息进行空间上的关联，以便确定攻击是否可成功，这样可信度值（reliability）就必须为变量，对于那些含有不可能存在端口安全事件的，则直接丢弃，图 4-33 展示了 Reliability 参数在 OSSIM 关联规则的变化过程。

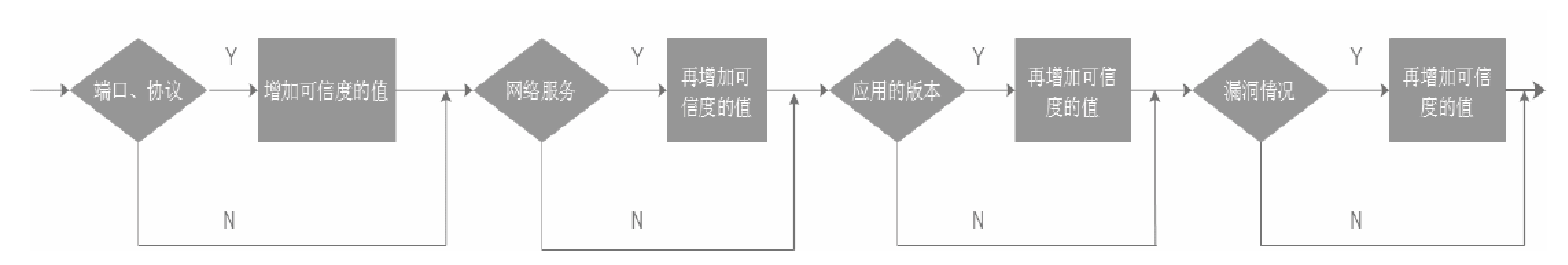


图 4-33　Reliability 参数变化过程

4.4.4　查看 SIEM 不同事件

在第 2 章讲解 SIEM 控制台时分析过不同日志的显示问题，这里再举个例子，比如 W32.Sasser.Worm 是一个尝试探测 MS04-011 漏洞的蠕虫（如 Microsoft 安全公告 MS04-011 文件中介绍）。它会由扫描随机选取的 IP 地址，进而传染含有漏洞的系统，如图 4-34 所示。

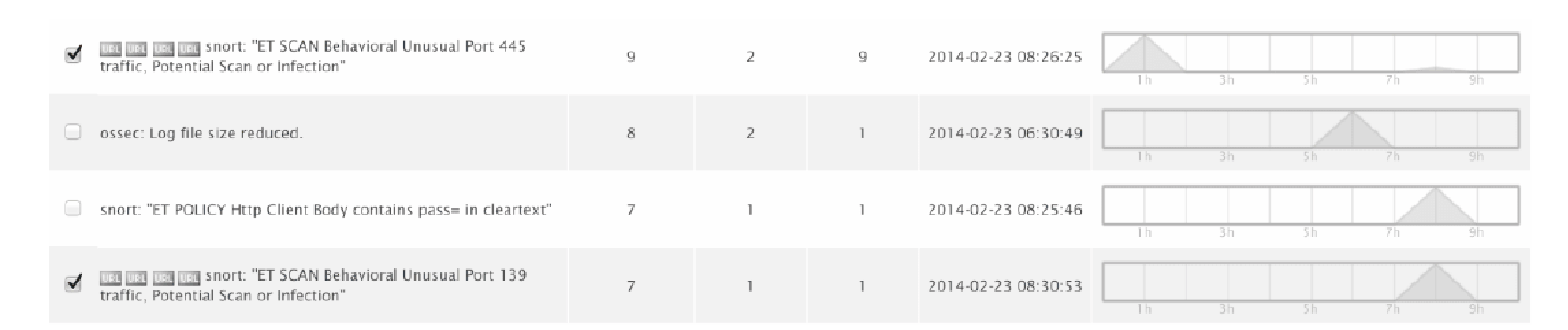


图 4-34　查看 SIEM 不同事件

因此 Risk 的取值范围为 0～10，值越高越要引起关注。一个资产的可靠性如何推断出来？是由资产关联和操作系统类型、端口、协议、服务名称以及版本综合判断。例如，一台服务器安装 Redhat Linux 9，且启动了 Apache 服务器，版本为 1.3.33，服务端口 80，协议为 TCP，此时 OSSIM 根据这些信息就能判断其可靠性，目前来看这样配置的服务器可靠性系统会达到 100%，也就是存在非常大的隐患。

再比如，运行 RPC 服务的 UNIX 服务器被冲击波蠕虫所攻击，这个攻击本身是危险的，它曾危害成千上万台主机并易于传播，但是它并不仅对该台服务器造成威胁，由于它利用了一

个严重的安全漏洞，因此它具有较高的优先级。

在实际操作 OSSIM 系统时，会通过 Intelligence→Correlation Directives 下 Directives 选项，新建 Directive 来设置优先级和可靠性，如图 4-35 所示。

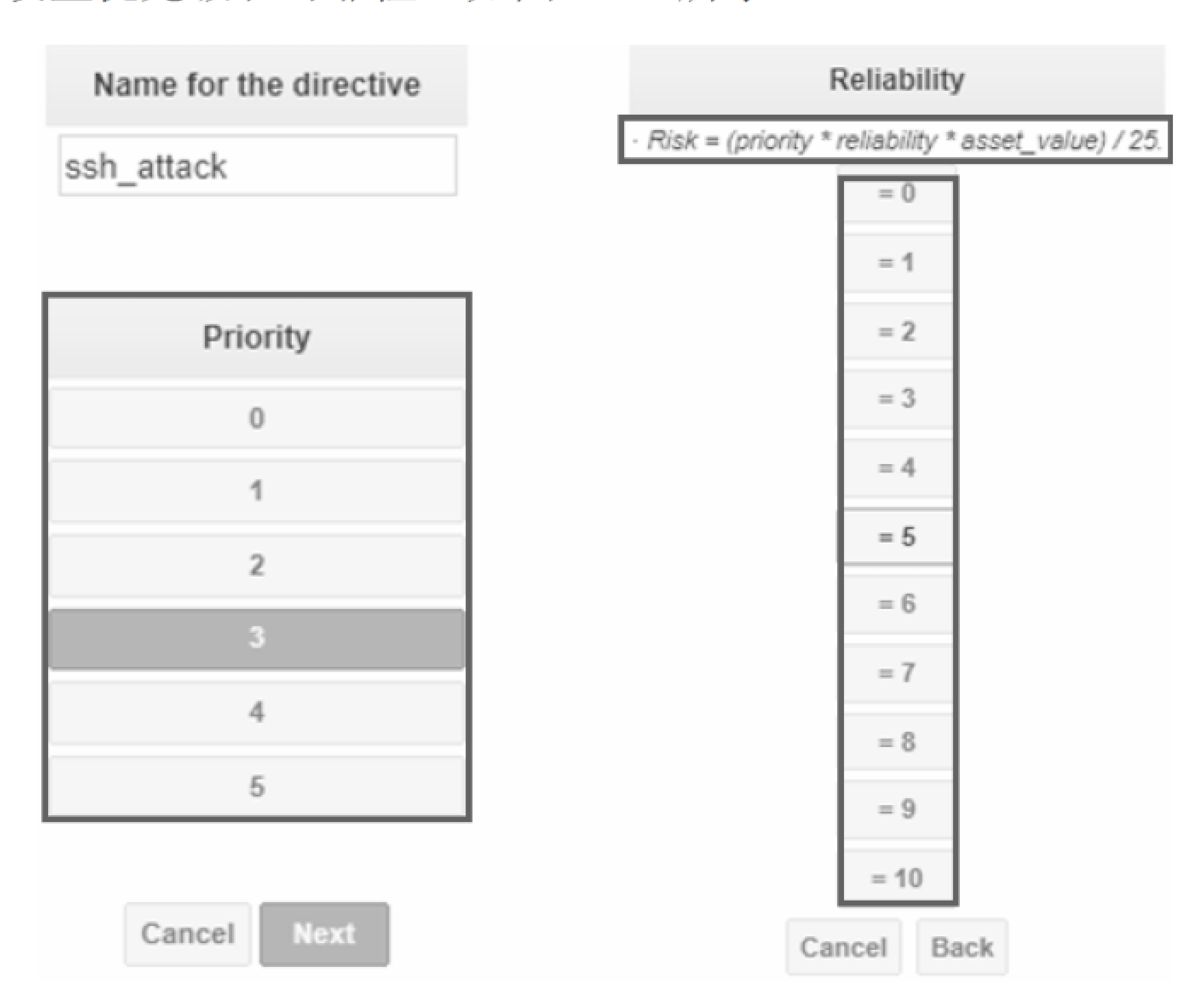


图 4-35　自定义优先级和可靠性

对于 OSSIM 4.8 系统中默认的数据源优先级和可靠性也可以调整，具体位置在 Configuration→Threat Intelligence→Data Source。首先，选中左边的数据源 ID，然后会显示如图 4-36 所示画面。另外在查询 SIEM 日志时，还可以将其加入到数据源数据库中。

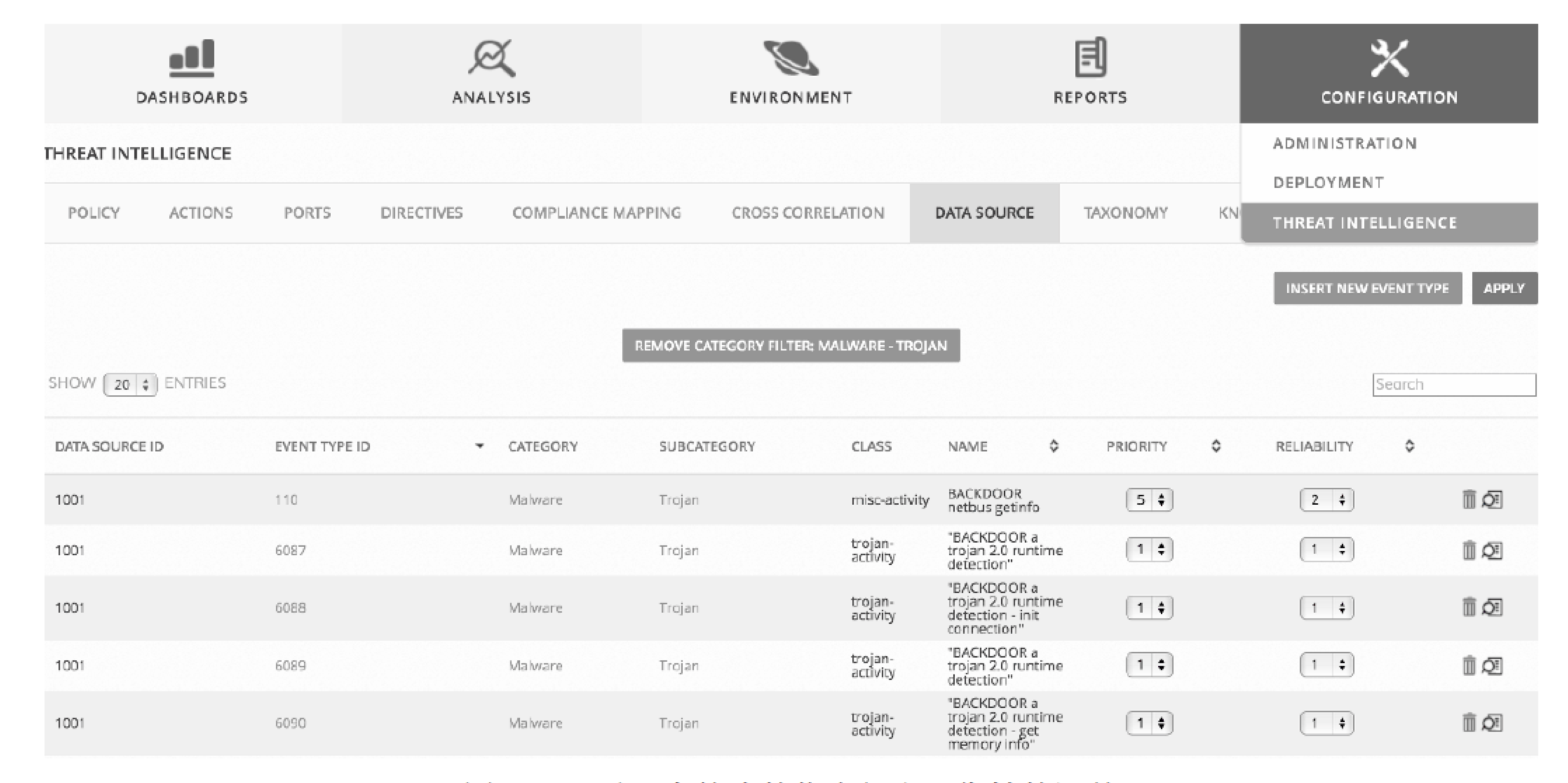


图 4-36　对现有策略的优先级和可靠性的调整

在 OSSIM 系统中，如果一个资产的风险值≥1，则关联引擎将 Alert 升级为 Alarm 并发出

告警，同时为区别其他低风险，会用其他颜色标记。换句话说，如果我们在前端 Web 界面上发现 Alarm 占多数，就要仔细查找原因，大家在系统中发现 Alarm 要引起高度注意，极有可能为入侵行为。

如果控制台在远端（分布式的情况），那么 Alert 将通过 E-mail 通知给管理员（admin 用户）。下面如图 4-37 所示，通过 Web 界面来看看如何快速找出 Alarm。Alert 在 Syslog 协议中定义为紧急消息，我们知道 Snort 具有实时报警功能，可以将报警发送到日志，Alert 在 Snort 规则中应用广泛。

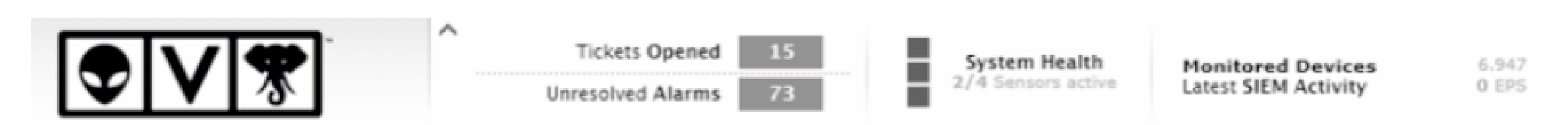


图 4-37　健康程度告警信息

在开源版 OSSIM 中会出现 AV-FREE-FEED 类报警，它们代表 AlienVault 公司免费的规则。

4.5 OSSIM 系统风险度量方法

4.5.1 风险判定

信息安全风险是威胁利用脆弱点对一个信息资产造成损失的可能性。通常，信息安全风险值的大小与信息资产的价值、面临的威胁及其具有的脆弱性密切相关。在 OSSIM 系统中还采用了一种叫做 CALM 的关联方法，所谓 CALM 的全名为 Compromise and Attack Level Monitor，该算法通过积累事件，并定时恢复风险值来获得当前的安全状态。每台主机都有一个 Risk_A 值和 Risk_C 值，它们含义如下：

- Risk_A：表示该对象正在被攻击；A 表示 level of attack，它意味着检测到了某攻击，但是对于此攻击成功与否并不能确定。
- Risk_C：表示该对象已被攻击；C 表示 level of compromise，compromise 原意是妥协，这里代表系统受到危害（或已受损害）。所以 C 的含义为某攻击已成功。

在 OSSIM 的 Alarm 报警中显示“System Compromise”，代表系统遭受攻击，需要管理员马上响应，这些攻击数量、频率、IP 地址、端口、关联等级和时间等信息被记录到 C 值。在系统中 C、A 的值越大，表示系统风险越高。

在 OSSIM 系统中，通过 Dashboards→Risk→Risk Metrics 查看 Riskmeter 可以掌握全局的 C/A 情况。在 OSSIM 4.8 版本中直接单击 Dashboards→Overview 下的 Threat Level 圆形图像（默认为 low，再往后发展依次变为 Precaution（警惕）、Elevated（提高）、High、Very High）

即可打开 Risk Meter，如图 4-38、4-39 所示。

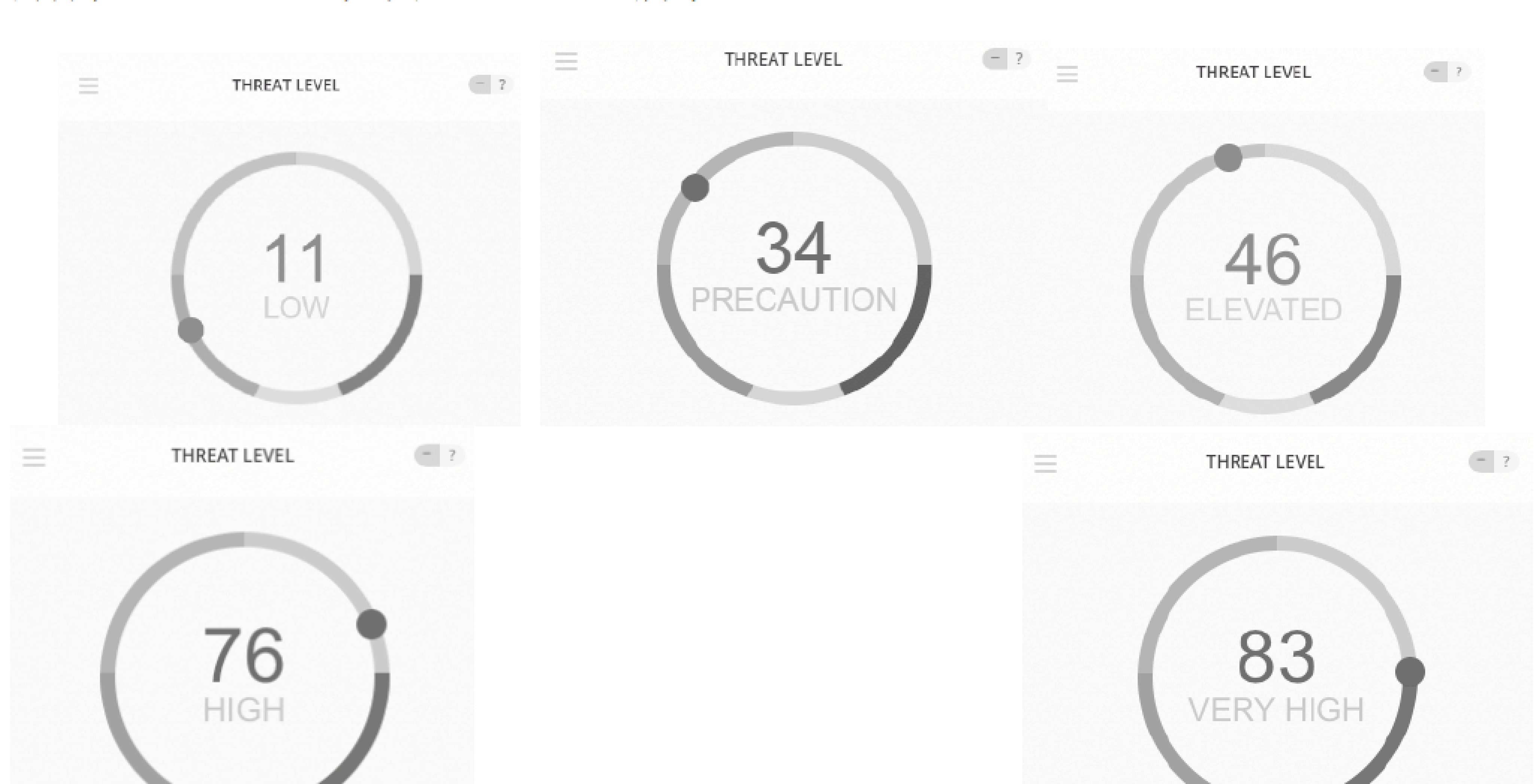


图 4-38　威胁级别的变化

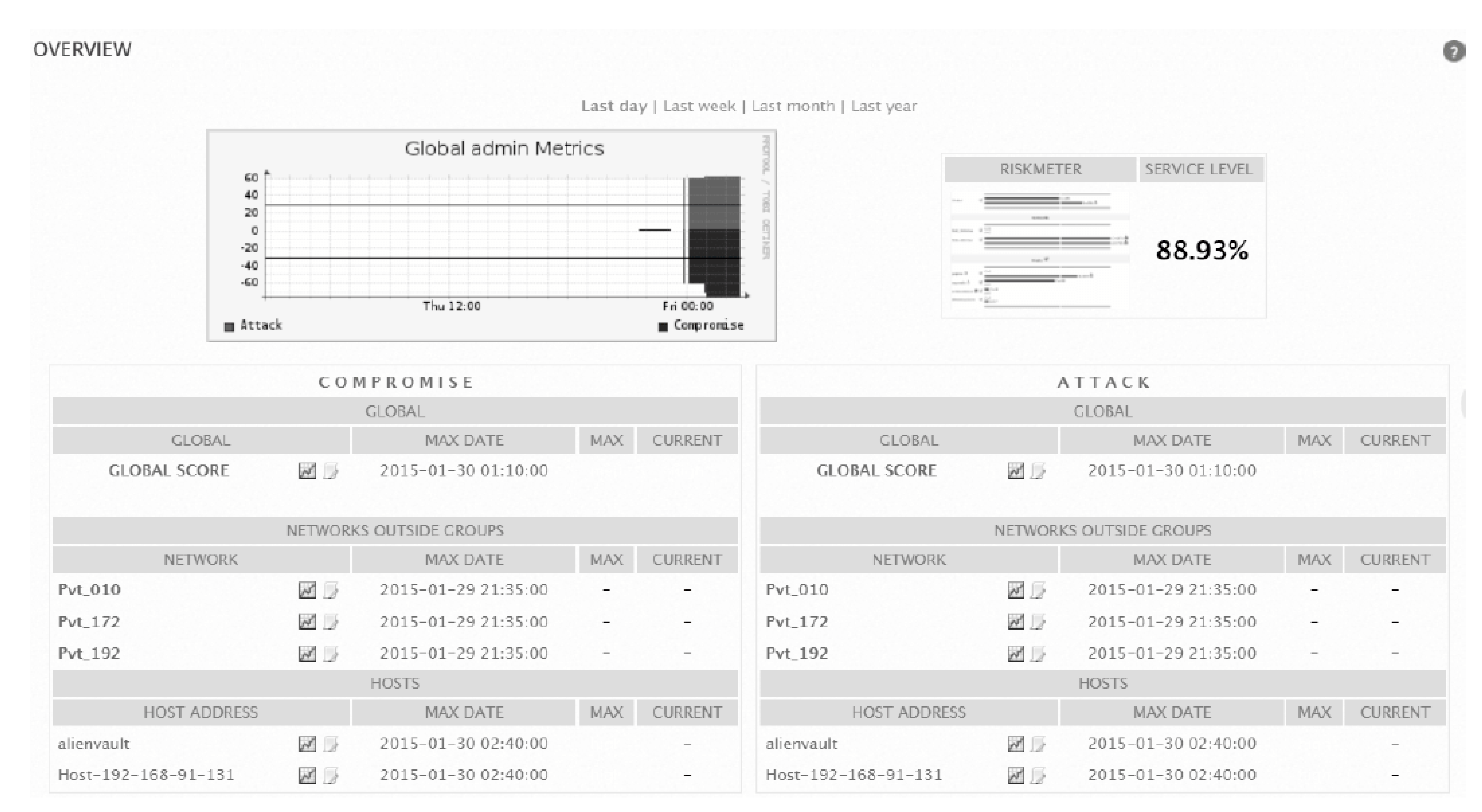


图 4-39　Riskmeter 右侧的 Service Level 参数

大家可能注意 Riskmeter 右侧的 Service Level 参数，它形象地表示出网络的健康值，理想状态下为 100%，随着网络中含有漏洞主机和服务数量的增加，这个值就会不断下降。

前面介绍过动态可信度值的问题，这里系统中处理 C、A 的值也是采用动态方式，它会定时恢复，管理员只关心最近一段时间内的风险状况，只对最近的安全事件感兴趣，而不用考虑很久以前的事件，所以需要引入定时恢复的机制，在 OSSIM 中引入时间间隔和恢复值，每隔

一个时间间隔（系统 10 秒）所有对象的 Risk_A 和 Risk _C 都会自动减 1。最终 Thread level 值会随着威胁数量的减少而自动降低，如图 4-40 所示。

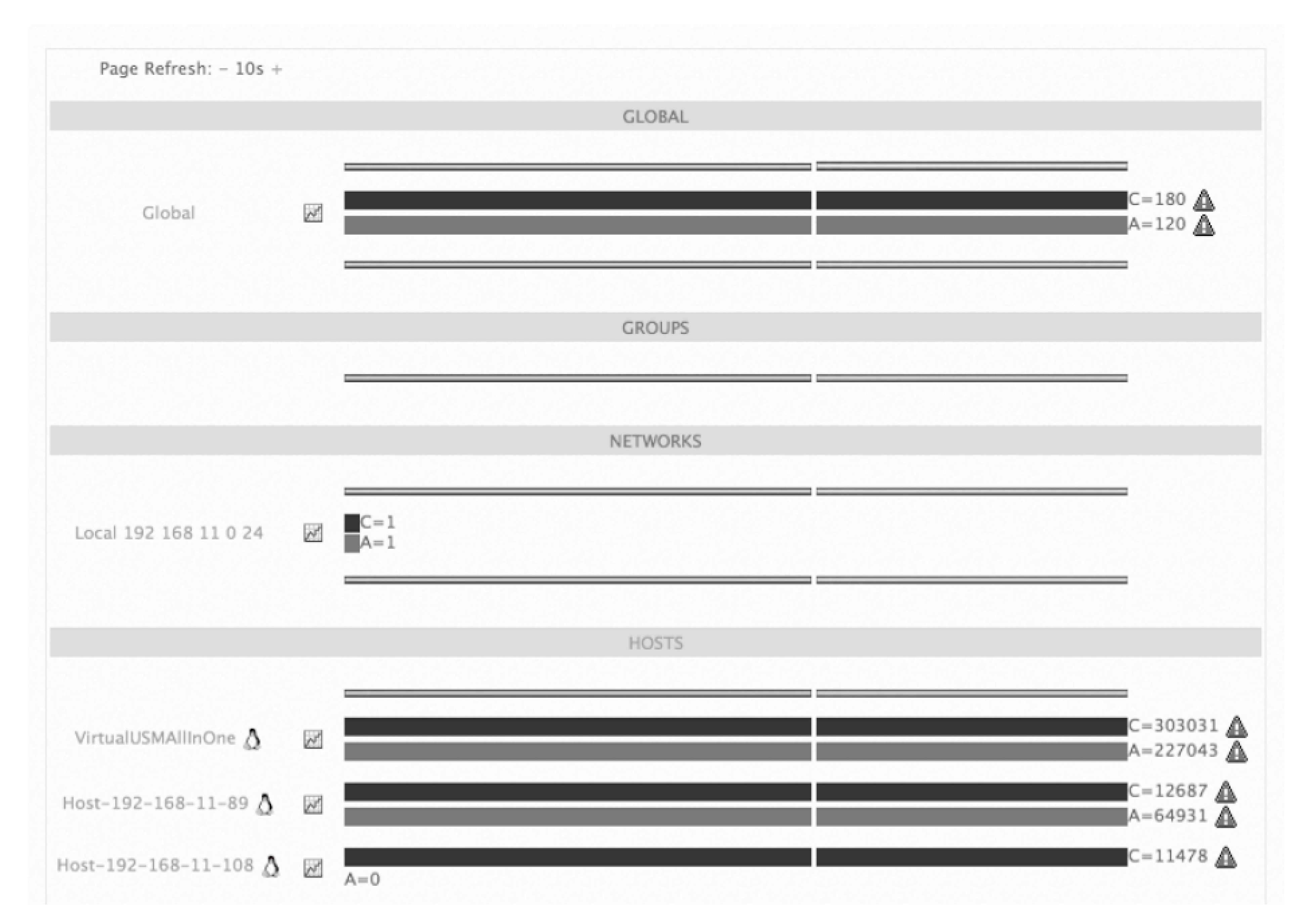


图 4-40　查询系统风险度量值

在 SIEM 控制台下，当鼠标悬停在 RISK 位置，能显示出 C、A 的值，箭头代表了方向，如图 4-41 所示。

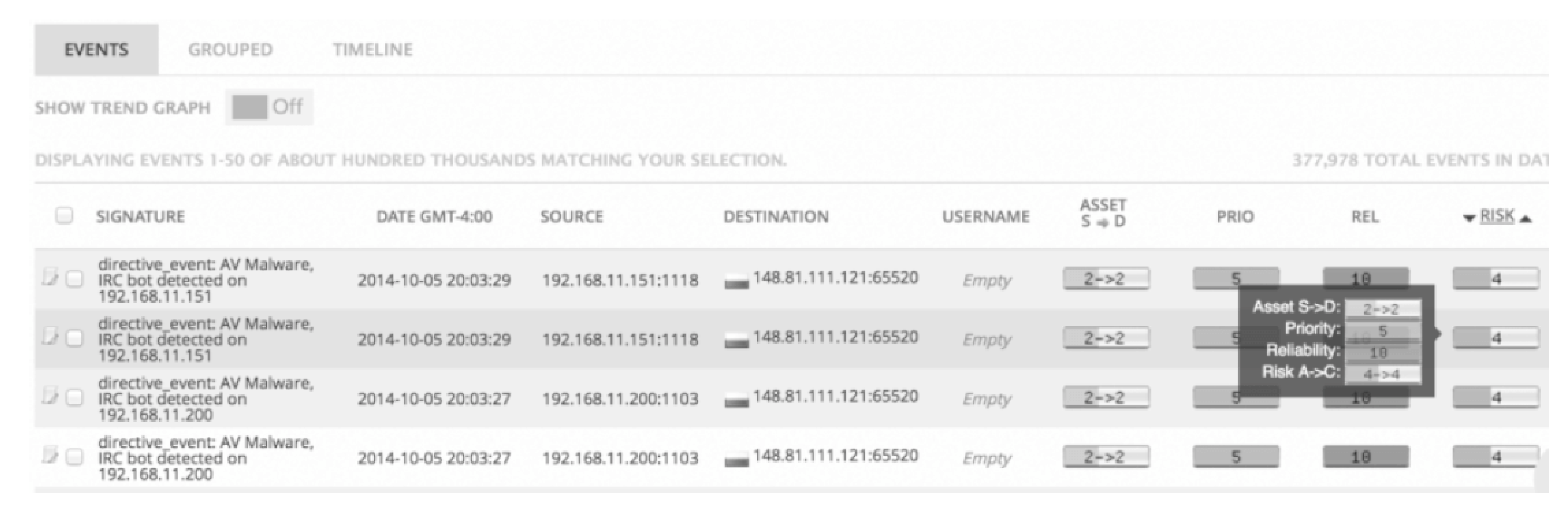


图 4-41　SIEM 下反映出的 C、A 情况

4.5.2　事件积累过程

每台主机都有 Risk_A 和 Risk_C，那么当 OSSIM 服务器接收到 Sensor 传来某主机的 Event 后，如果该事件是从主机 A 到主机 B，那么就增加 A 的 Risk_C 和 B 的 Risk_A 值，如图 4-42 所示。

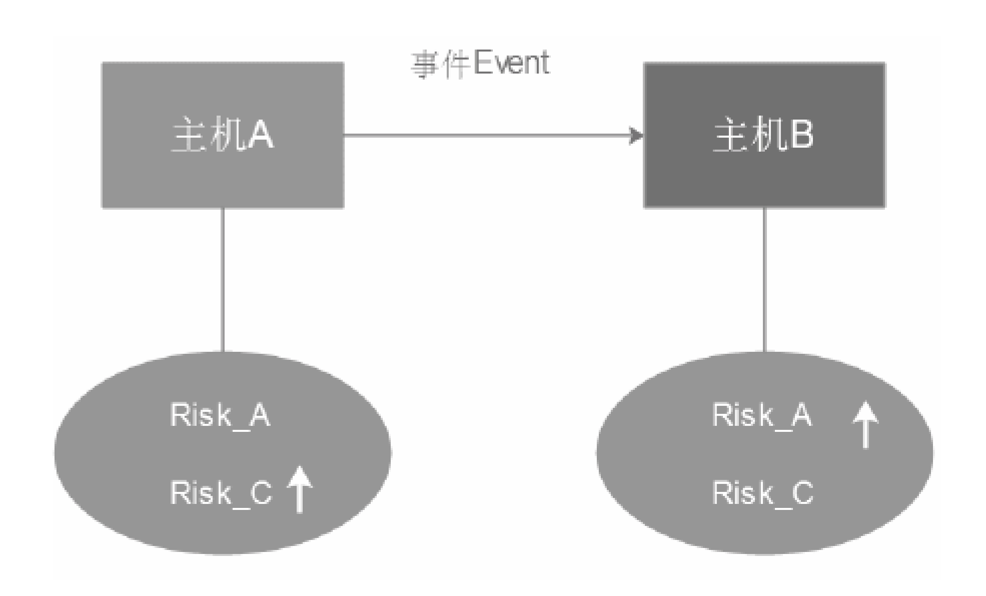


图 4-42 风险值变化

接着，如果该事件属于攻击响应，主机 A 和 B 的 Risk_C 都增加。那么仅从 Risk_C 和 Risk_A 的数值上看，数值越高，风险越大。所以经过启发式关联分析后，系统就得到一个综合安全指示状态，系统默认为 Low，再向后发展依次为 Precaution（警惕）、Elevated（提高）、High 和 Very high。

在关联分析中，对于基于事件的关联分析，它需要定义攻击场景，所以只能检测已知攻击，对于未知攻击则无法检测，如何解决这种问题呢？OSSIM 采用前面介绍的 CALM 分析的方法来检测未知攻击，它使用了类似于温度计的风险指示器（在新版本 OSSIM 使用了速度表的仪表盘）

基于 CALM 的启发式关联，以大量事件为输入的前提下，提供一个实时的 C、A 值输出图表，实时反映了整个网络的安全状况。我们在首次对监控网段的设备进行资源发现时，可以为资源池里的设备设置 C、A 的临界值，方法是在 Environment→Assets→Asset Discovery 右侧资源池中输入一个待监控网段，然后选“Start Scan”按钮，此时系统开始扫描并记录下发现的设备，待扫描完成会列出设备清单，此时单击“Update Database Values”按钮，目的是更新数据库，到这一步就可以开始设置临界值了，比如 C、A 默认设定为 30，如图 4-43 所示。

图 4-43 C、A 值设置

4.6 OSSIM 中的关联分类

事件关联（Event Correlation）将大量的安全事件过滤、压缩、归一化处理后，再提取最重要的安全事件。我们知道内、外部入侵攻击行为，都有逻辑上联系。黑客的异常行为往往分为若干步骤，每个步骤都会在不同设备和系统上留下蛛丝马迹，将这些事件关联处理，就可将所有的相关信息相互关联，从而发现安全隐患。

4.6.1 关联分类

OSSIM 的关联主要分为以下几类：

- 交叉关联（Cross Correlation）：是指事件与目标漏洞之间的关联。交叉关联通过将入侵检测系统 Snort 产生的告警信息与漏洞扫描工具 Nessus 规则进行交叉关联，综合评估威胁程度的方法，在 OSSIM 平台也是将 snort 产生的告警信息与 Nessus 漏洞扫描报告进行关联分析。
- 资产清单关联（Inventory Correlation）：事件与目标特性之间的关联（包括操作系统关联、端口关联、协议关联、网络服务名关联等）。前提条件是资产清单中需要包含有资产的特征信息，比如 Snort 报告某个机器出现 Linux 漏洞的攻击警报，但实际是台 Sun 服务器，那么 OSSIM 就可以通过关联资产清单中的操作系统特征，鉴定这条 Snort 的报警是误报，当然策略可由管理员来制定。

OSSIM 的关联引擎在系统中扮演着重要的角色，首先我们先找到其位置，对于 OSSIM 4.6 系统位置为 Configuration→Threat Intelligence→Cross Correlation。

下面是查看具体规则，例如 EXPLOIT SSH server banner overflow（SSH Server 远程缓冲区溢出漏洞）的关联规则，如图 4-44 所示。

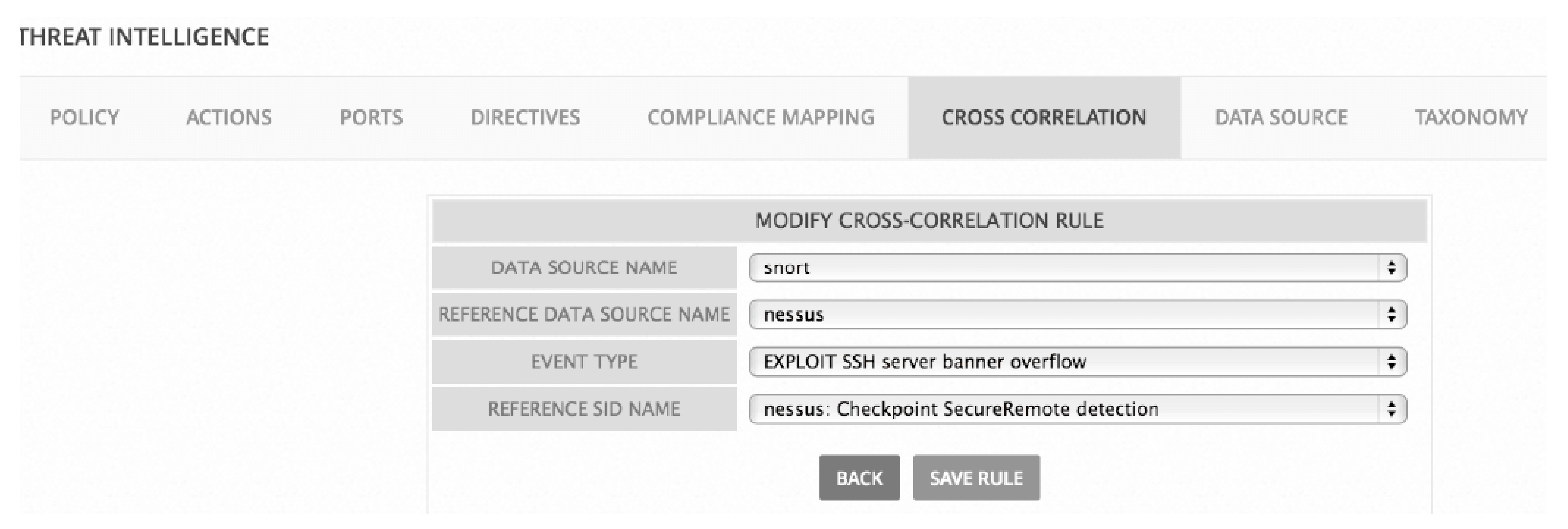


图 4-44　交叉关联规则

数据源来源于检测插件，图中“EVENT TYPE”数量达到 2 万余种，所以显示比较慢，其信息来源于/etc/snort/sid-msg.map，由/usr/share/ossim/scripts/create_sidmap.pl 脚本程序负责调用

到 OSSIM-db 中，而 gid-msg.map 这个文件内容主要和 Snort 规则有关。在"Data Source Name"和"Reference Data Source Name"的选项中都提供了各种插件，如图 4-45 所示。

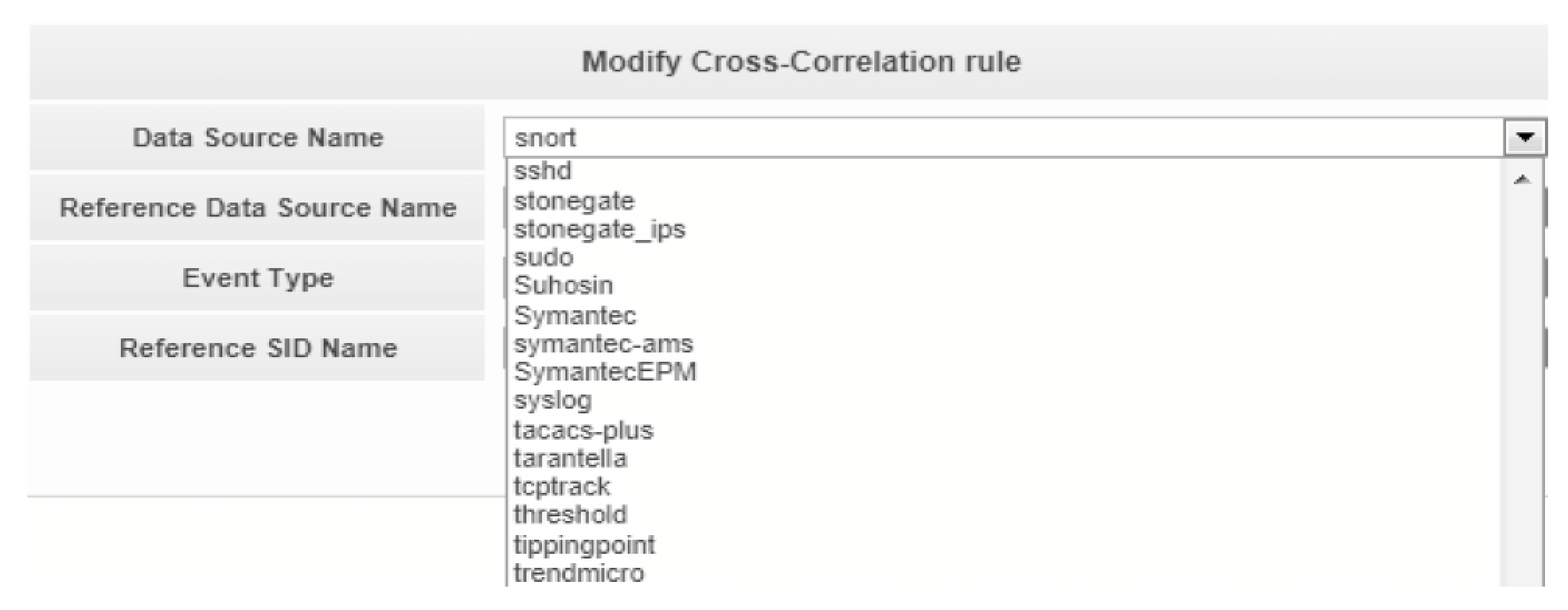


图 4-45　丰富的数据源及插件

4.6.2　关联指令分类

下面通过关联指令分类表，可以清楚了解到 OSSIM 适合检测到那些攻击类型，具体内容如表 4-3 所示。

表 4-3　攻击检测关联指令分类说明

分类	解释	举例
用户自定义 User Contributed	由用户创建/修改的指令，默认为空	由用户创建
攻击指令 AlienVault Attacks	这一类包括指令用于对各种应用漏洞进行攻击检测	AV Attacks, Successful OpenSSL HeartBeat attack AV Service attack, successful WebDAV service exploitation on DST_IP AV Service attack, successful FTP service exploitation on DST_IP AV Service attack, Microsoft SQL attack via web server against DST_IP (sp_adduser) AV Web attack, web shell backdoor detected AV Web attack, SQL injection attacks detected against DST_IP AV Web attack, XSS attacks detected against DST_IP AV Attack, file /etc/passwd access on DST_IP
暴力破解 AlienVault BruteForce	这类指令包括了对一些需要认证的服务，例如 SSH 进行暴力破解攻击	AV Bruteforce attack, SSH authentication attack against DST_IP (destination IP) AV Bruteforce attack, Windows authentication attack against DST_IP AV Bruteforce attack, Windows authentication attack against DST_IP (Snare) AV Bruteforce attack, HTTP authentication attack against SRC_IP AV Bruteforce attack, FTP authentication attack against SRC_IP AV Bruteforce attack, HTTP authentication attack against SRC_IP AV Bruteforce attack, FTP authentication attack against SRC_IP AV Bruteforce attack, VNC authentication attack against SRC_IP

（续表）

分类	解释	举例
拒绝服务攻击 AlienVault DoS	该类指令用于检测各种应用和服务的拒绝服务攻击	AV Service attack, successful denial of service against IIS web server on DST_IP（MS07-041） AV Service attack, successful denial of service against Microsoft RDP service on DST_IP AV Attacks, Possible DDoS using SQL servers AV Service attack, successful denial of service against Samba server on DST_IP AV Service attack, successful denial of service against MySQL server on DST_IP（CVE-2009-0819） AV Service attack, successful denial of service against DST_IP（MS06-035） AV Service attack, successful denial of service against Microsoft FTP Service on DST_IP（MS01-026）
AlienVault Malware	这类指令用于检测恶意软件攻击	AV Malware, botnet Koobface activity detected on SRC_IP（source IP） AV Malware, spyware VaccineKillerIU detected on SRC_IP AV Malware, trojan Swizzor detected on SRC_IP AV Malware, DDoS trojan G-Bot detected on SRC_IP AV Malware, IRC bot trojan IRCBrute detected on SRC_IP AV Malware, mobile trojan HongTouTou detected on SRC_IP AV Malware, worm Slammer propagation attempt against DST_IP AV Trojan Backdoor.Win32.Agent.bjjv - Operation Hangover detected on SRC_IP AV Malware, PlugX DNS CC
AlienVault Misc	这类指令用于检测不同于上述的其他类型攻击行为	AV Misc, suspicious executable download from a dynamic domain on SRC_IP AV Misc, EXE file download from a Dynamic DNS host 这类规则仅十几条
AlienVault Network	这类指令用于检测网络异常攻击	AV Network attack, too many dropped inbound packets from DST_IP AV Network attack, SYN flood detected on SRC_IP AV Network attack, UDP flood against DST_IP AV Network attack, IP spoofing detected on SRC_IP AV Network attack, port security violation on SRC_IP
AlienVault Policy	这类指令用来检测违反策略的行为	AV Policy violation, vulnerable Java version detected on SRC_IP
AlienVault Scada	该类指令用于检测 SCADA 系统	AV SCADA attack, Modbus scanning or fingerprinting against DST_IP，这类规则仅 10 条

（续表）

分类	解释	举例
AlienVault Scan	该类指令用于检测网络的各种扫描活动	AV Network scan, Nmap scan against DST_IP AV Network scan, local host scanning port 1433/TCP on SRC_IP AV Network scan, SSH service discovery activity from SRC_IP AV Network scan, Wikto web vulnerability assessment tool usage against DST_IP AV Network scan, SNMP default community string discovery detected from SRC_IP AV Network scan, Microsoft Remote Desktop outbound scanning behaviour detected from SRC_IP AV Network scan, host with worm scanning behavior on SRC_IP AV Network scan, OpenVAS vulnerability assessment tool usage against DST_IP

4.6.3　指令组成

事件中每条指令有全局属性以及一个和多个规则、指令信息以及赋予该指令对应的知识库。下面先看一条指令的主要属性，其结构如图 4-46 所示。

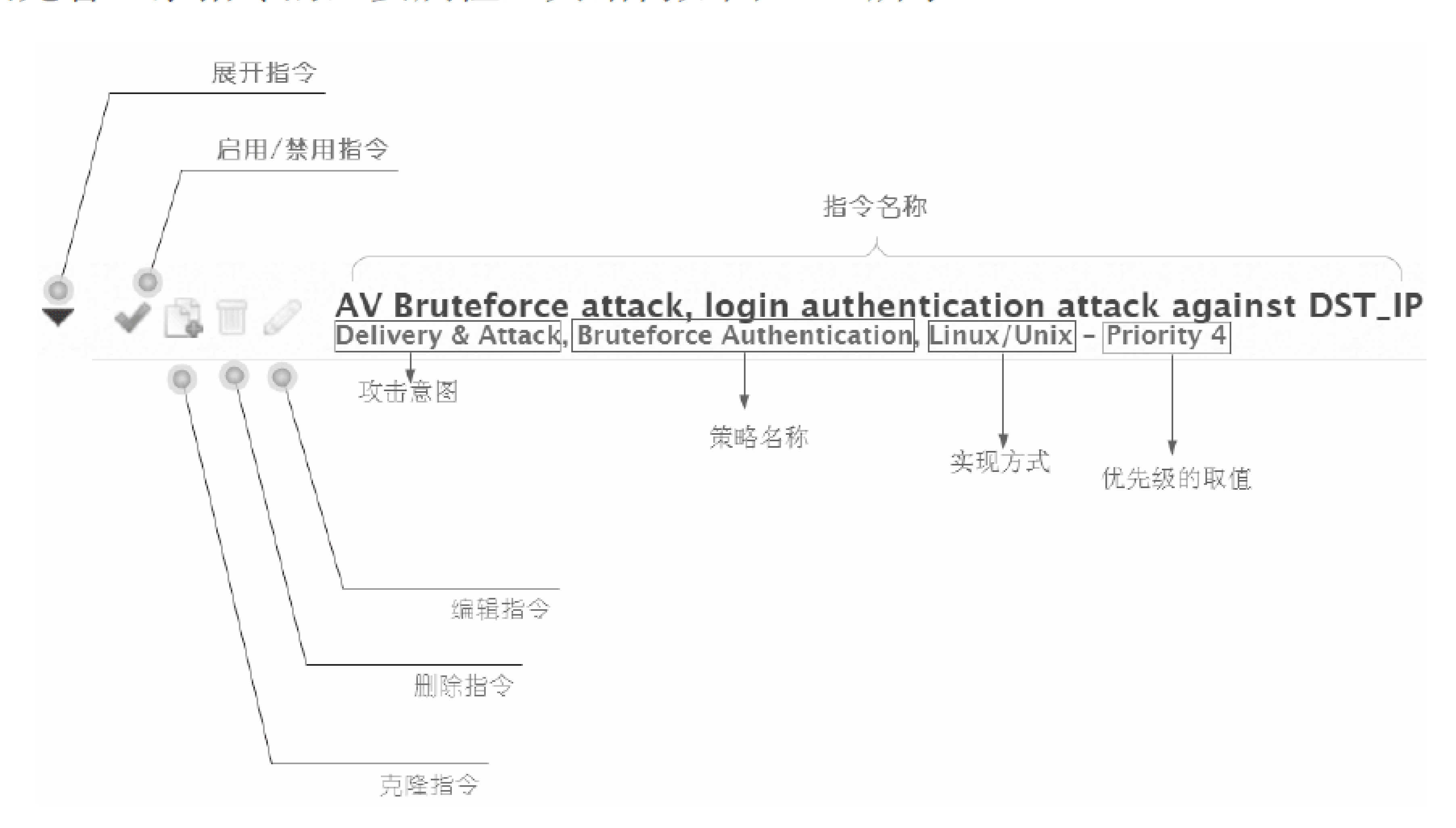


图 4-46　一条指令的全局属性和管理选项

每条关联指令包含以下全局属性：

（1）ID：代表指令的 ID 号，这是唯一的标识符，但指令的 ID 并不在 Web 界面中显示。

（2）Name：事件或报警名称。

（3）Intent、Strategy 和 Method：攻击意图、策略、方法。该项主要用于指令分类。这三项内容在新建指令过程中就能发现，比如这三项也体现在 Web UI 界面的 Alarm 告警示例中，如图 4-47 所示。

图 4-47　Alarm 中显示 INTENT、STATEGY 和 Method

（4）Priority 定义了对检测到攻击的影响，用于风险评估计算。前面讲过，由指令生成的所有事件都有自己的优先级，设置为指令的优先级值。在关联指令中能显示优先级的大小是 USM5.0 之后提供的新功能。指令种类说明如下：

- 展开指令：该按钮可以展开指令中的规则。
- 启用/禁用指令，启用指令用 表示，禁用指令用 表示。每次操作会有“The directive was successfully disabled/enabled”提示，此时重启服务的按钮会被标记为红色，提示用户重启关联引擎，以便生效。
- 克隆指令，系统内置的指令默认已调试完毕，如果用户想改动设置，首先要克隆该指令，然后在对克隆的指令进行修改。克隆的指令会保存到用户分类中，而且克隆的指令并不包含知识库，对于知识库需要手动添加，如图 4-48 所示。

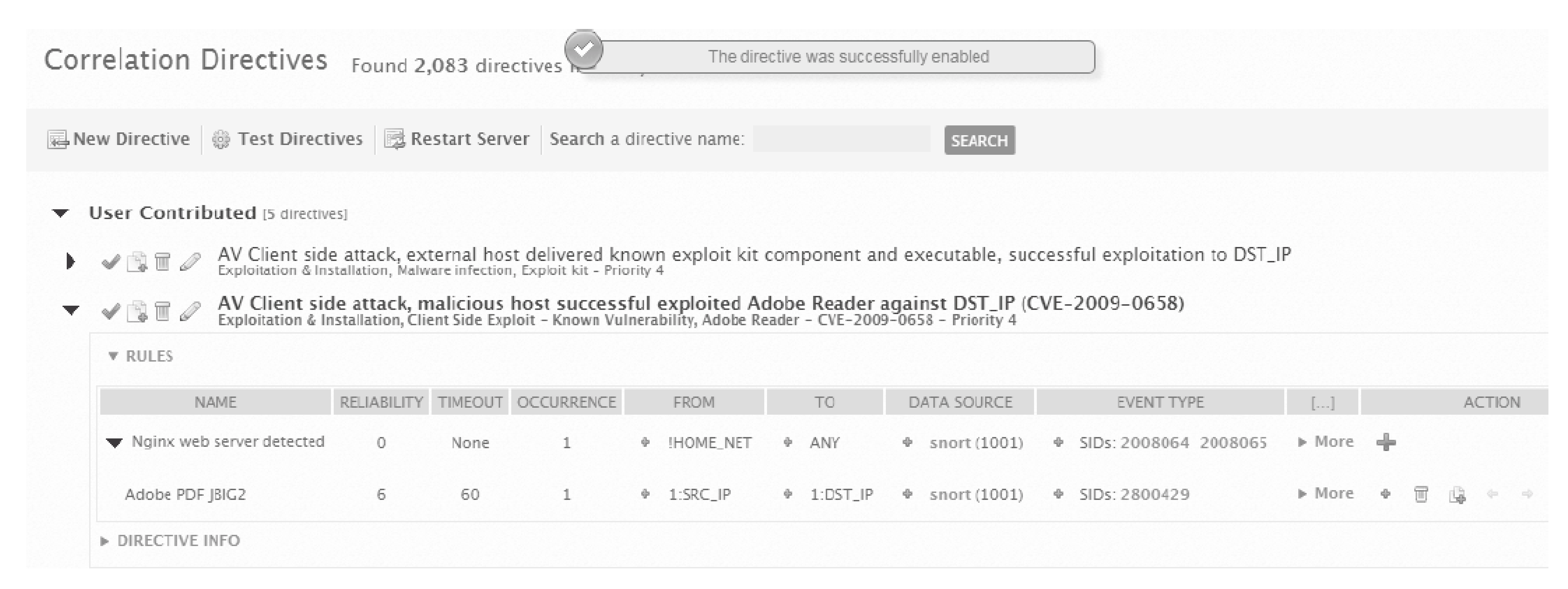


图 4-48　克隆规则

待克隆指令调试通过之后，这时方可将原来的指令禁用。

- 删除指令：注意系统默认的指令不能在 Web UI 中删除，图标显示灰色 ，而用户

自定义或克隆的指令才允许被删除，此时图标显示为 。

- 编辑指令：注意系统默认的指令不能在 Web UI 中编辑，图标显示灰色 ，而用户自定义或克隆的指令才允许被编辑，此时图标显示 。

我们对指令的增、删、改都是对于 User Contributed 指令和 Clone 指令而言，系统默认指令规则只能启用或禁用。所以对于自己修改的规则位于/etc/ossim/server/UUID/user.xml 文件中。对于 UUID 的值，每台机器都不同，使用中需要注意。

4.6.4　读懂指令规则

在图 4-49 中，显示了系统内置的 AV Network attack 指令，由 3 个关联级别和 4 条规则组成了这条指令。有很多从 DST_IP 来的入站数据包会被丢弃，数据源来自于 cisco-pix。

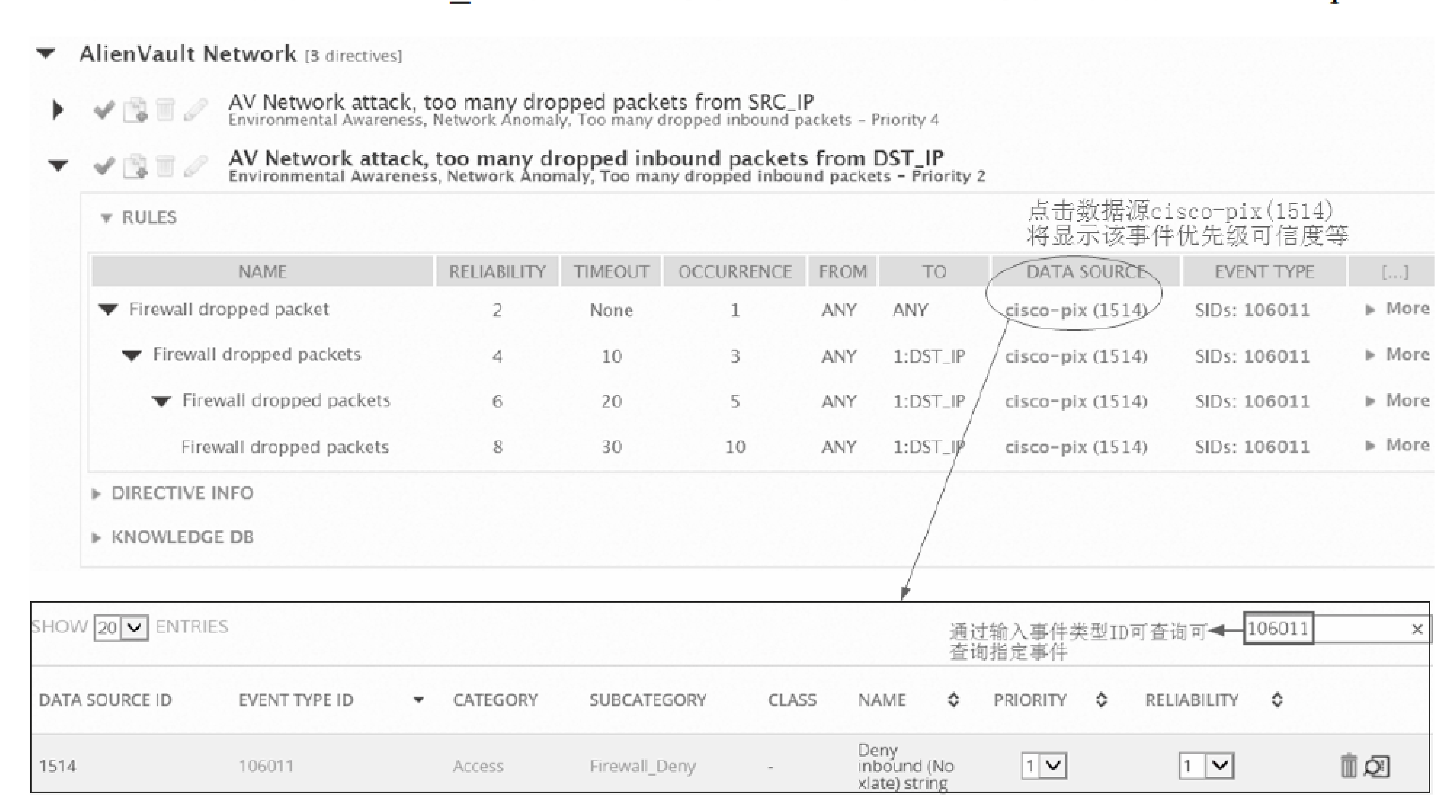


图 4-49　查看优先级和可信度

我们了解一下工作过程，我们知道 RELIABILITY 默认值为 2，当和指令的第 1 个规则相匹配时，“Deny inbound string”的事件 SID 为 106011，这是由 Cisco Pix 数据源插件负责识别。

在第 2 条规则里，参数 OCCURRENCE=3，TIMEOUT=10，说明如果在 10 秒钟内，发生 3 条这样的事件，而且都发往相同目的地，该指令的 RELIABILITY 将会变为 4。

以此类推，再看第 3 条，如果在 20 秒内，有 5 个这样的事件发生，该指令事件的 RELIABILITY 将变为 6。

第 4 条规则，如果在 30 秒内，有 10 个这样的事件发生，该指令的 RELIABILITY 将增加到 8。

如果读者充分理解了这些规则的含义，以及自己要实现的目的，也可以单击数据源，手动修改初始的 Priority 和 Reliability，以达到检测要求。

4.6.5 Directive Info

指令信息这部分显示了 AlienVault 合规映射和报表，它列出了由指令而触发的报警信息，如图 4-50 所示，左边第一列，列出一些附加信息（称为属性）的指令，如什么样的攻击指令是用来检测。网络异常属性设置为“YES”。IMP 表示短暂的影响；QOS 表示服务质量，这些属性设置，用于 B&C 业务合规性报告，在 AlienVault USM 内置的报告中。

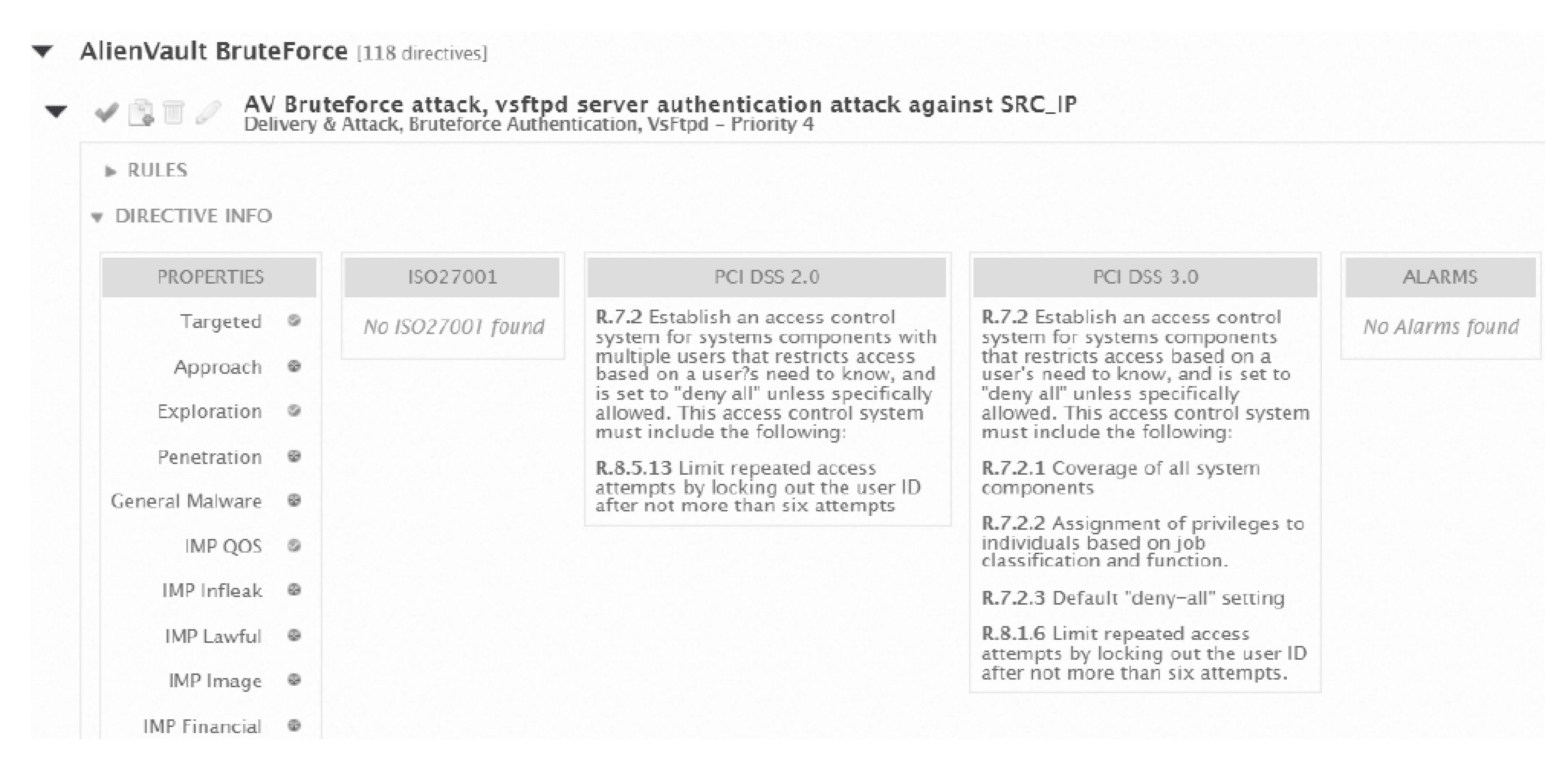


图 4-50　显示指令信息属性以及合规信息

如图 4-51 所示。在左侧 PROPERTIES 里显示的属性内容，右侧显示了匹配值，如果需要修改内容，单击左下方 EDIT 按钮。单击 REMOVE 按钮也可以清除所有属性。更多关于 ISO27001 和 PCI DSS 的内容大家参考第 9 章合规报表一节。

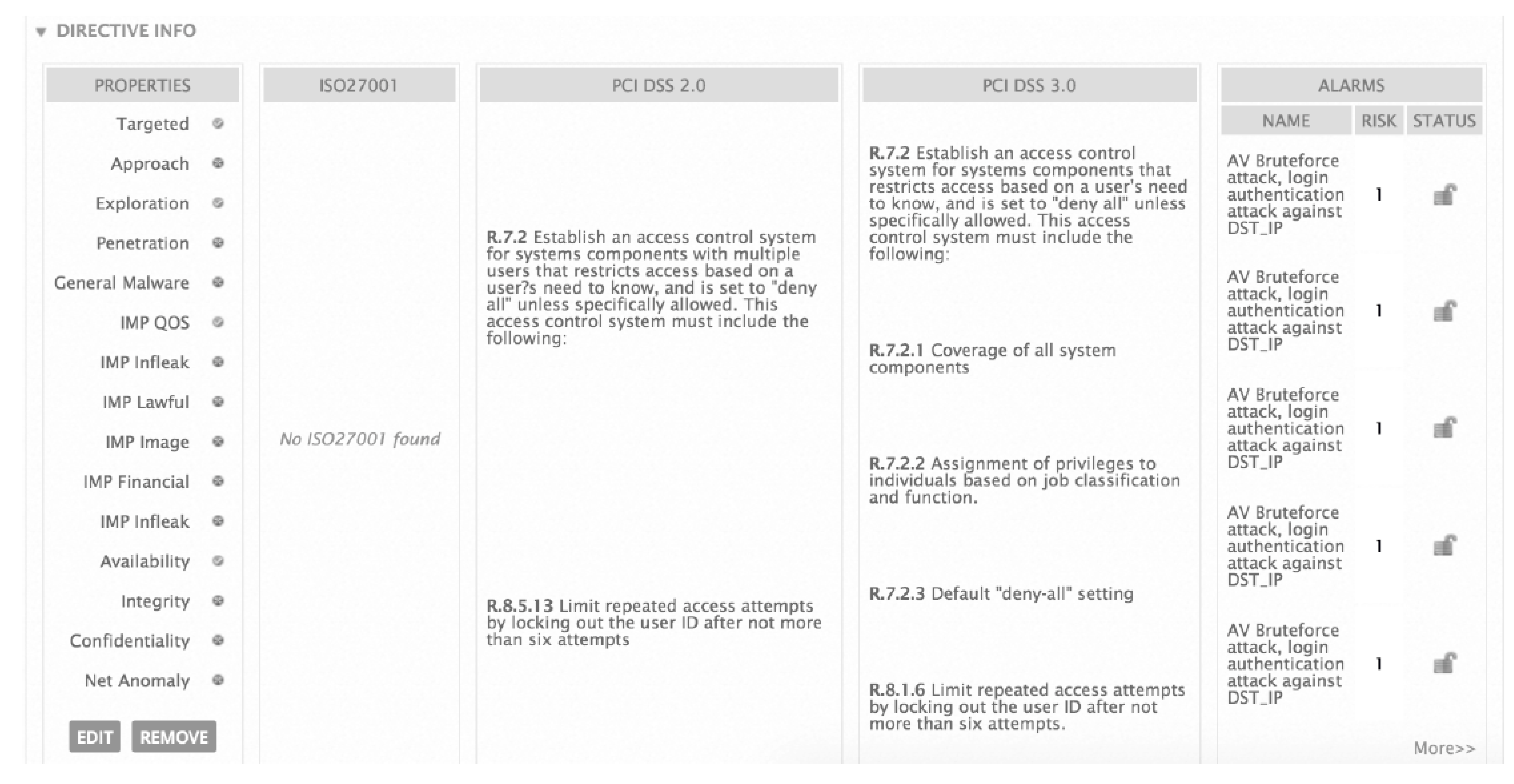


图 4-51　Directive Info 中显示 PCI DSS+Alarm

4.7 新建关联指令

关联分析过程复杂，但是在 OSSIM 提供的友好界面中实现并不复杂，在 OSSIM 4.8 系统中建立关联指令规则需要以下 7 个步骤：

（1）首先在 Threat Intelligence→Directives 中选择“New Directive”按钮，新建指令，如图 4-52 所示。

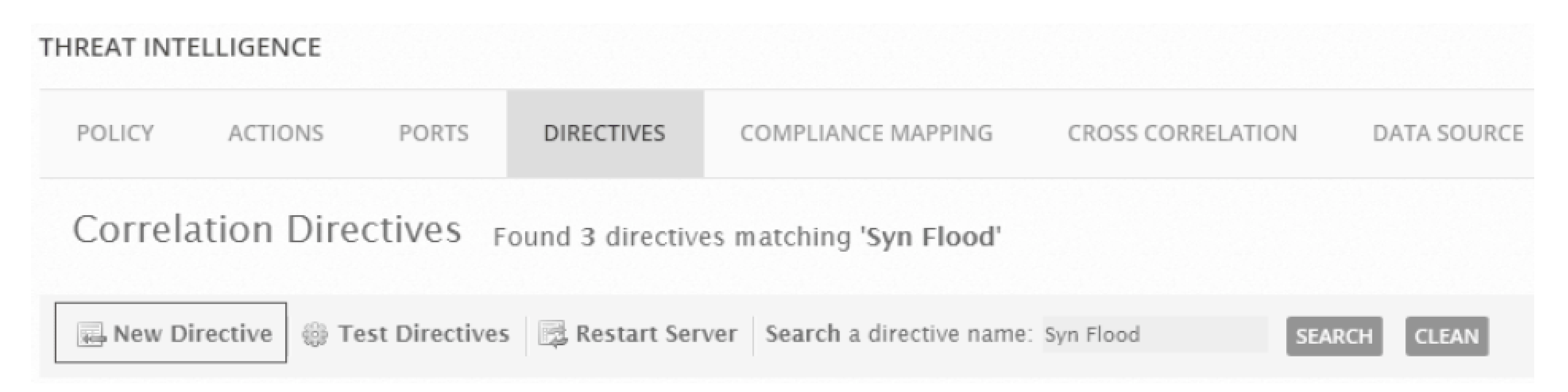


图 4-52　新建指令

（2）为新指令指定名称，选择攻击检测类型、分类、方式以及优先级，然后单击下一步，如图 4-53 所示。

（3）指定规则名称，如图 4-54 所示。

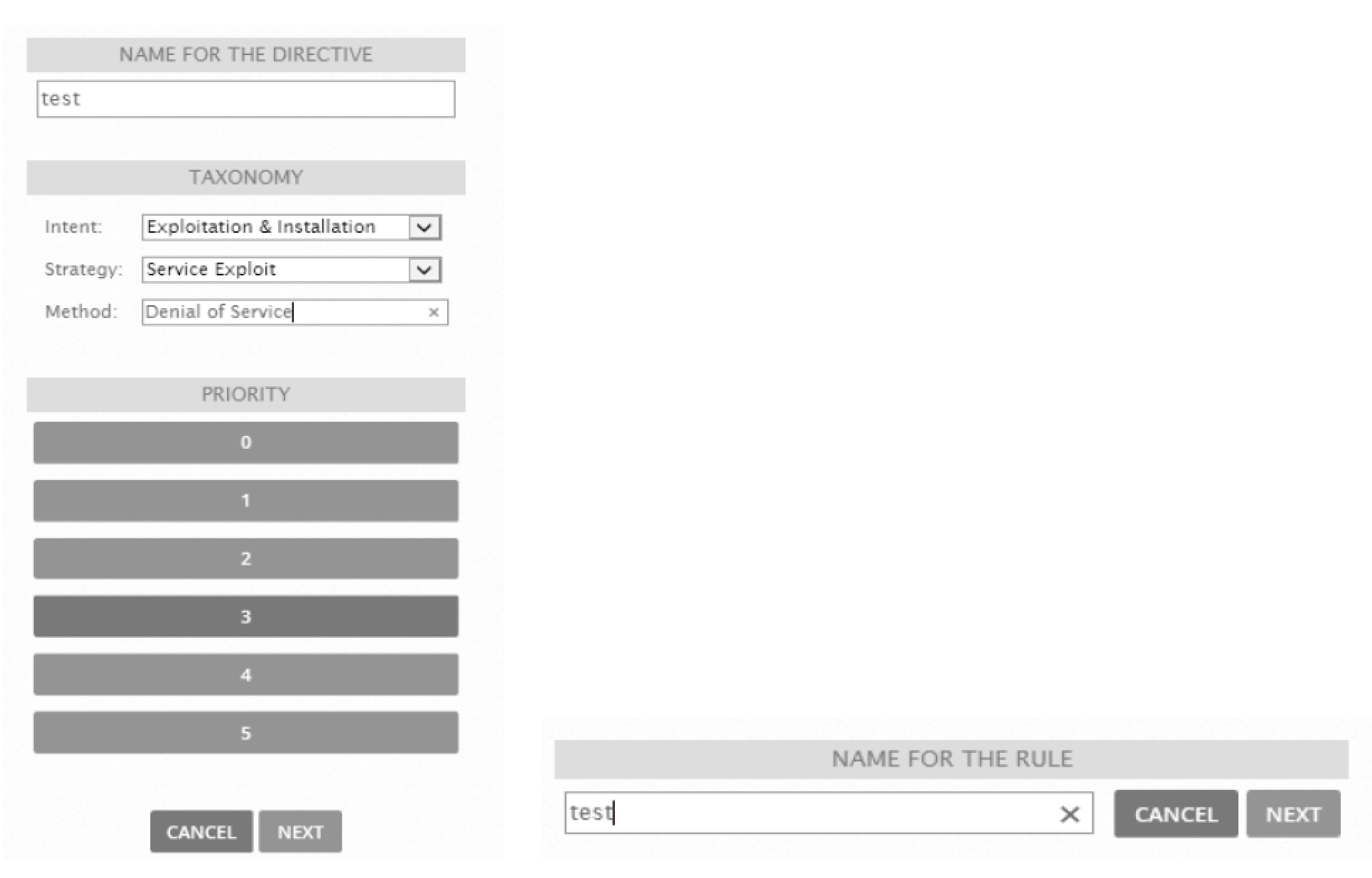


图 4-53　建立新指令　　　　图 4-54　规则名

（4）根据事件类型选择插件，比如选择 Apache 插件，如图 4-55 所示。

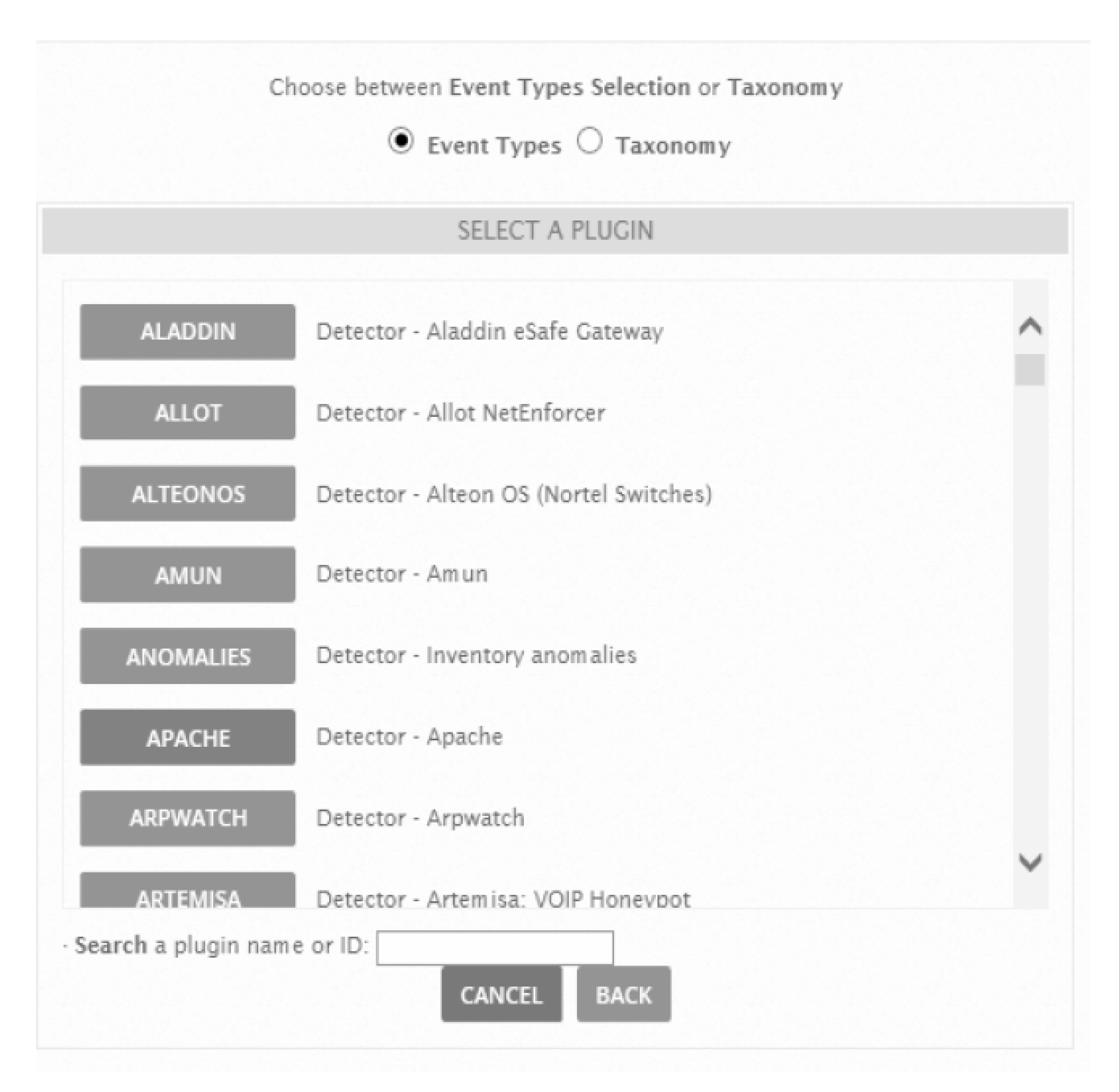


图 4-55　选择插件

（5）根据 Apache 插件规则的子类型，选择 Apache 返回代码，如图 4-56 所示。

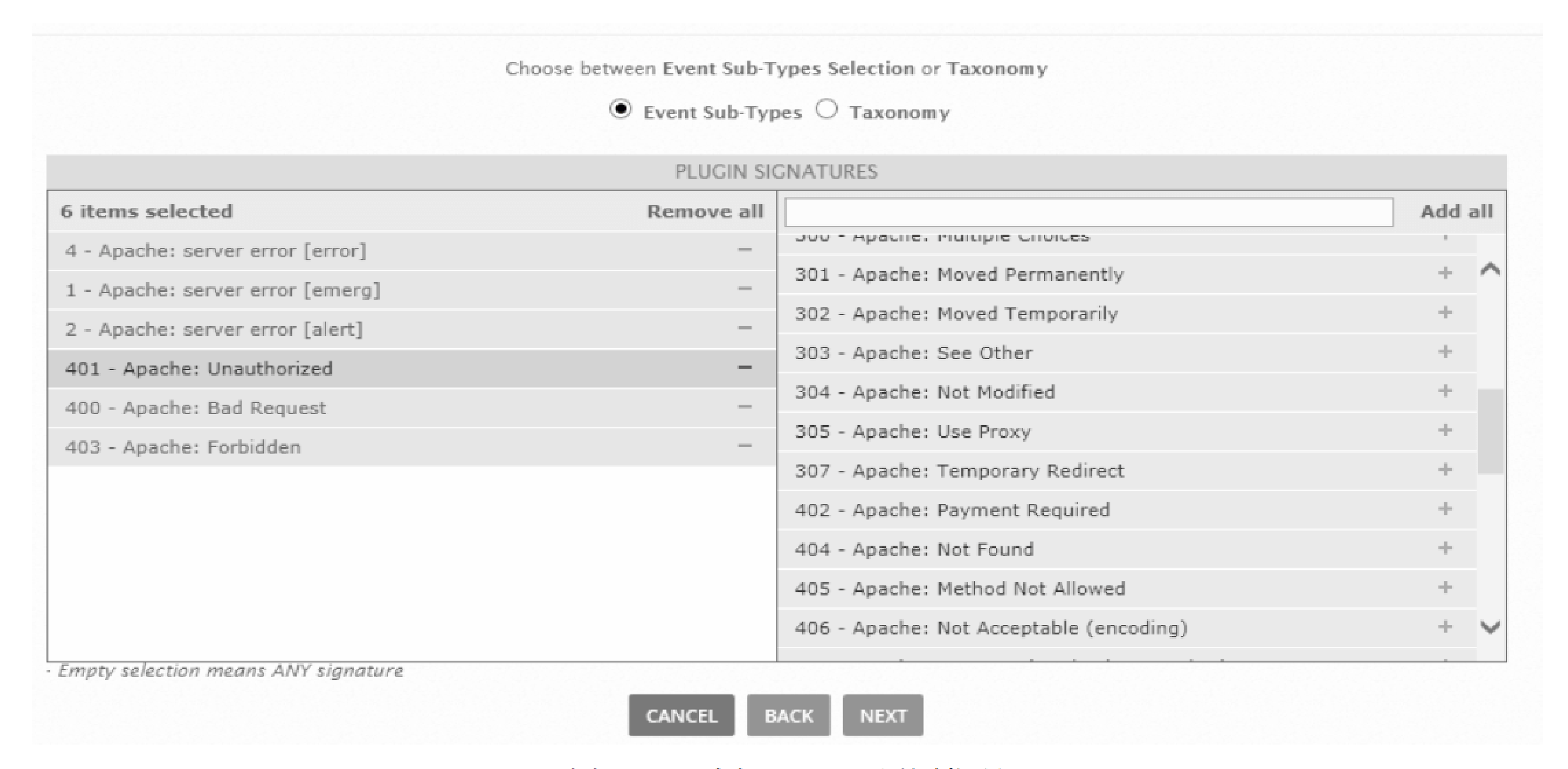


图 4-56　选择 Apache 返回代码

（6）选择源和目标主机（或网络），如图 4-57 所示。

图 4-57　选择主机和目标网络

（7）选择设备可靠性，如图 4-58 所示。

图 4-58　选择设备可靠性

（8）最后选择 FINISH 按钮，还可以指定附加规则，例如选择协议类型和传感器，如图 4-59 所示。

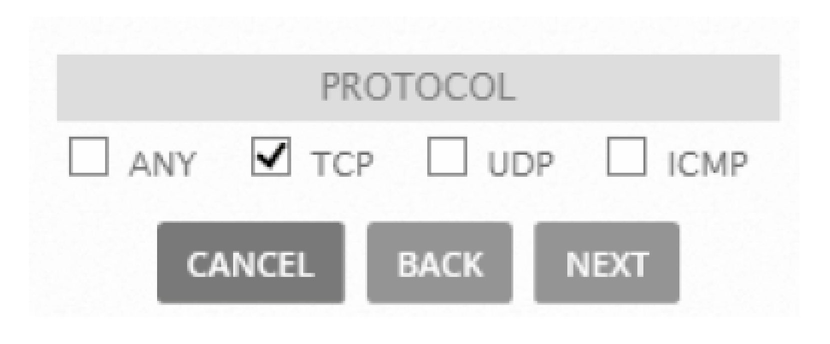


图 4-59　指定附加规则

当设置完关联指令后，接下来可以测试规则，例如系统中有条 scanning port 445/TCP 的关

联指令，通过它发现网络中存在蠕虫进行 445 端口扫描时，进行报警，如图 4-60、图 4-61 所示。

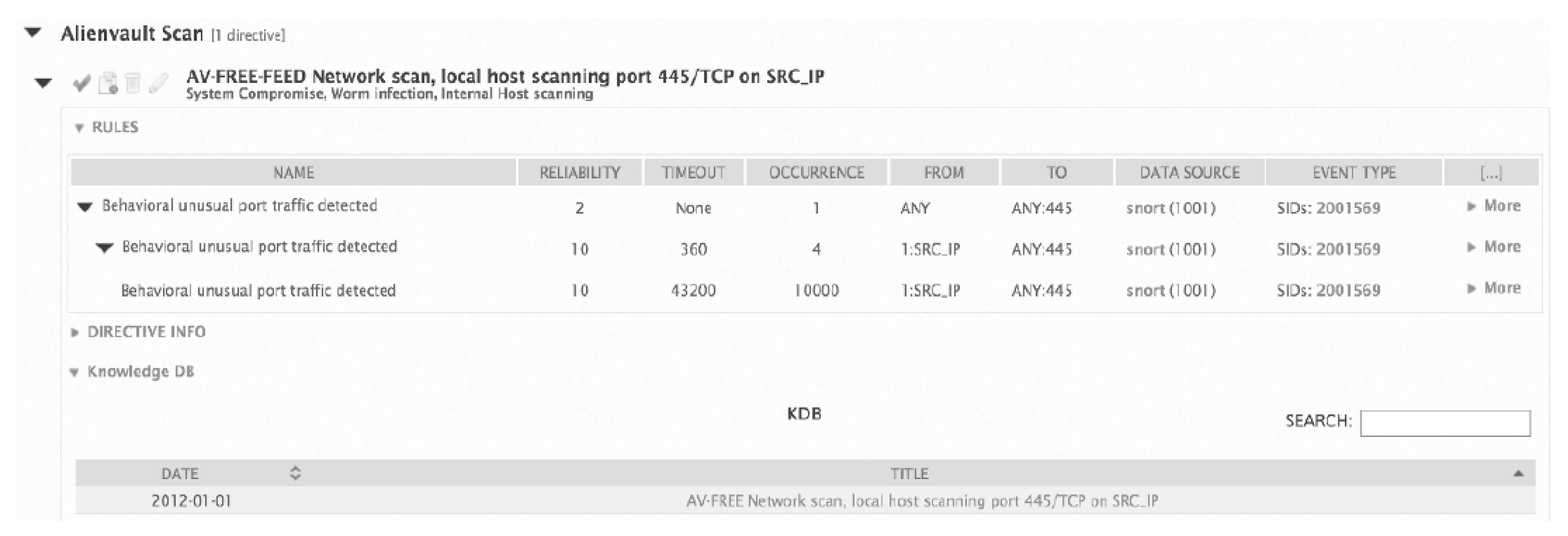


图 4-60　445 端口扫描的过滤规则

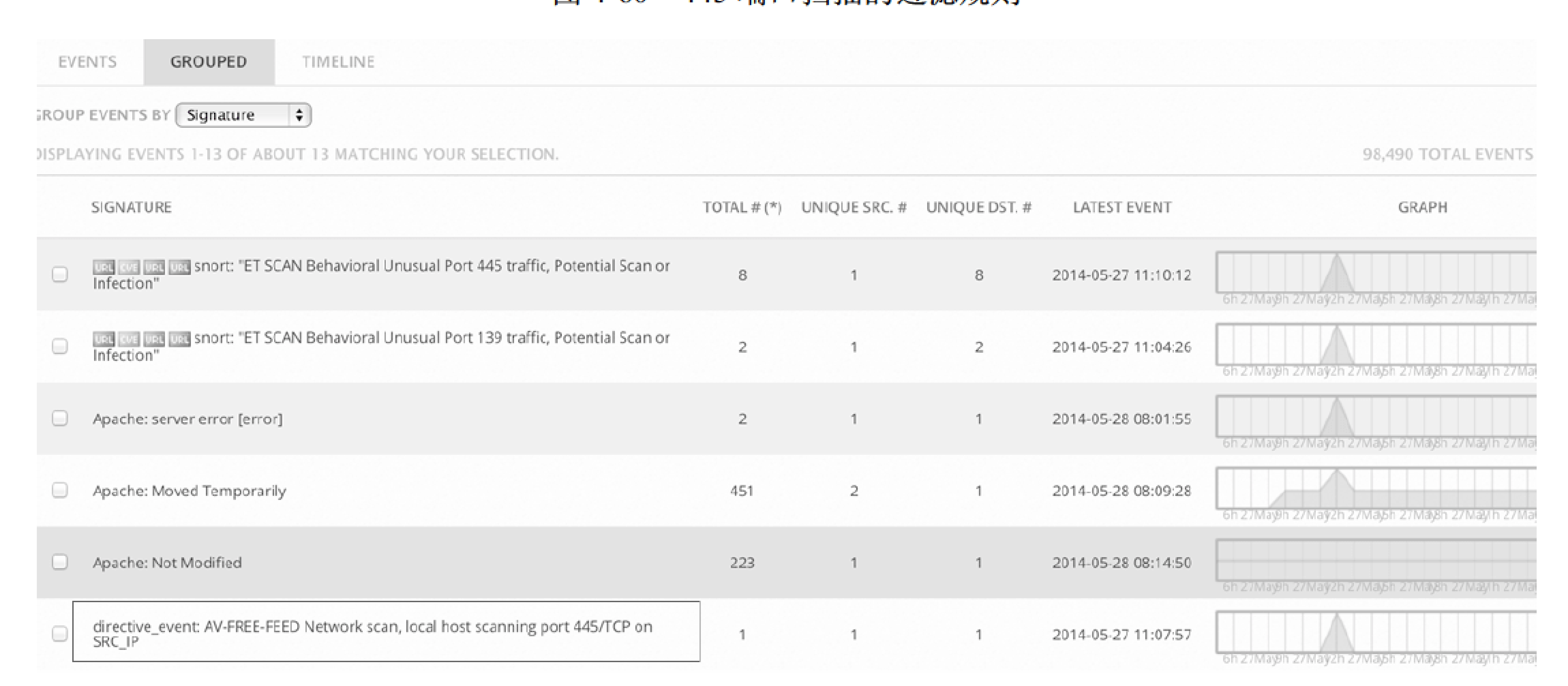


图 4-61　通过定义的指令发现扫描并在 SIEM 中报警

4.8　OSSIM 的关联规则

关联分析的核心通过一组关联指令来完成，下面利用 SSH 暴力破解的例子加以说明，SSH 暴力破解（Brute Force）大约自 UNIX 诞生之后，就衍生出来的一种攻击行为，据统计发现大约有 50%以上的用户名是 root。一个低可靠性（low reliability）的 SSH 服务器，在经过 100 次登录尝试之后成功登录，而高可靠性（high reliability）的 SSH 服务器，则经过 10000 次登录才成功，那么关联引擎可通过在一定时间内，登录的次数，以及这些时间的不同源地址和相同目标 IP 地址，以及这些 IP 地址来自于那些国家，它们信誉度如何等信息，来综合判定攻击以，便自动作出响应，如图 4-62 所示。

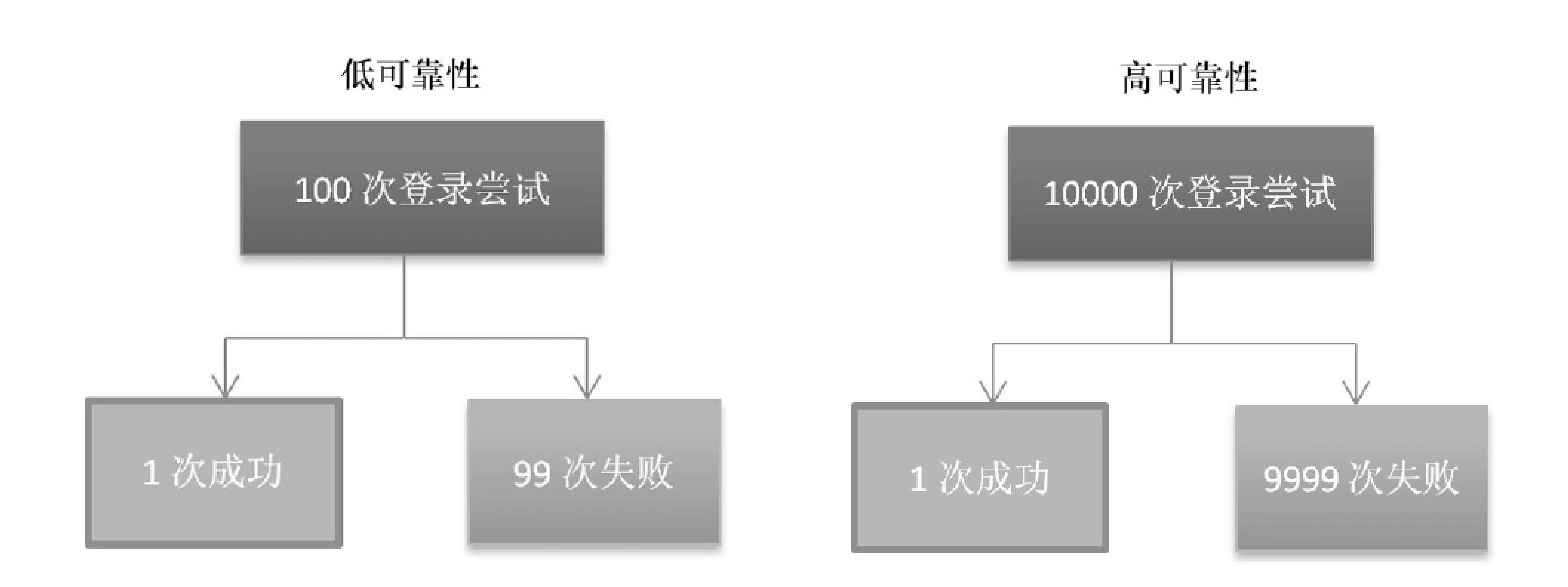


图 4-62　高低可靠性的比较

4.8.1　关联指令配置界面

在 OSSIM 中，关联指令的界面通过 Configuration→Threat Intelligence→Directives 打开，经过前面几节内容的学习，大家已经接触过关联指令的界面，下面我们归纳一下，如图 4-63 所示。

图 4-63　OSSIM 关联指令界面

界面上的关键功能解释如下：

- New Directive：单击此选项以从头开始创建一个新的指令。
- Test Directives：单击此按钮，将检测当前新建/修改的指令是否正确。
- Restart Server：单击此按钮，将重启 ossim-server 进程，凡是修改了任何一条关联指

令，都需重新启动 Server，按这个按钮比你重启整个 USM 服务器更明智。需要注意的是，虽然重启时间短暂，但在服务重启期间，所有关联数据会丢失。

下面以一条完整指令 Av Bruteforce attack,login authentication attack against DST_IP 为例，介绍其内部结构，如图 4-64 所示。

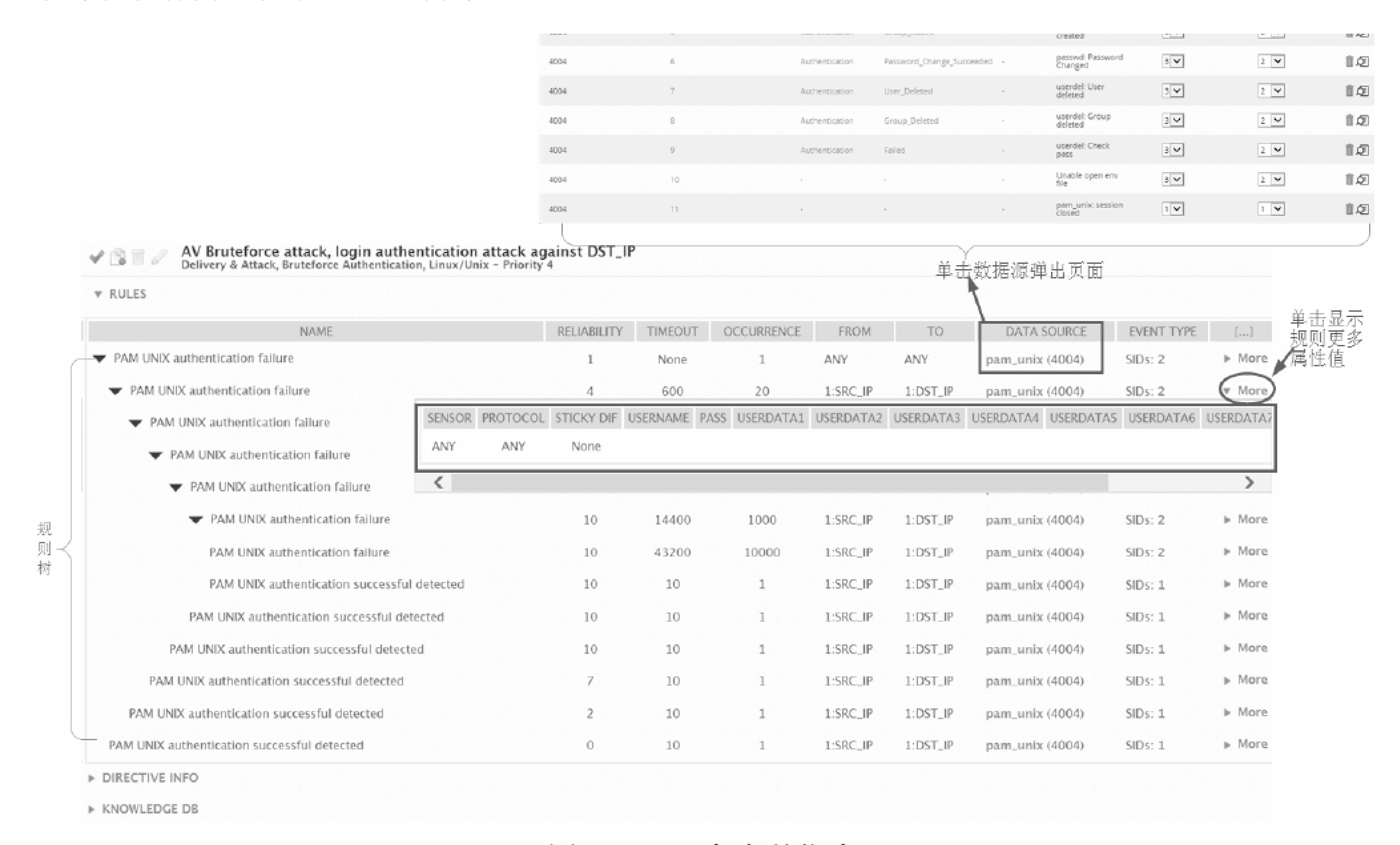


图 4-64　一条完整指令

- Sensor：传感器，用于收集各种事件信息。
- Protocol：协议，包括 ANY、TCP、UDP、ICMP。
- Sticky different：用于文件及目录属性设定的 sticky，从 OSSIM 4.3 起在关联指令中添加了新属性，Sticky different（不同的粘滞位）域规则中 Sticky different 为 None。它是用来设定指令规则的属性。

有关 Sticky different 的取值大家可参考/etc/ossim/server/directives.xsd 文件

```
<xs:simpleType name="stickydifferenttype">
... ...略
</xs:simpleType>
```

当新收集的事件输入到达关联引擎时，它将和已有的事件相关联，如果事件属性（如 IP 地址和端口号）相同，但一个指令的属性里可以设置不同的粘贴位，比如 None、DST_PORT、SRC_IP、PLUGIN_ID 等，用来区别收到的不同事件，以便进行相互关联。例如，在端口扫描类攻击中，如果设置目标端口为 Sticky different，那么只有来自不同目标端口的事件才被关联，在 Web UI 中效果如图 4-65 所示。

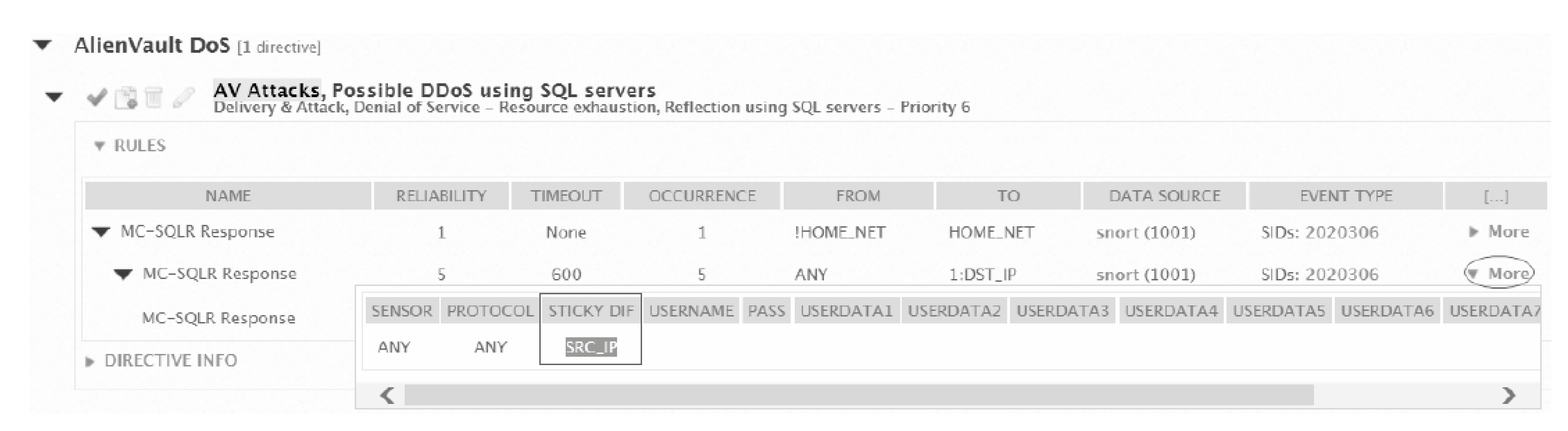


图 4-65　STICKY DIFF

打开/etc/ossim/server/alienvault-dos.xml 查看关联指令 AvAttacks、Possible DDoS，嵌套规则如图 4-66 所示。

```
<directive id="34047" name="AV Attacks, Possible DDoS using SQL servers" priorit
y="6">
    <rule type="detector" name="MC-SQLR Response" reliability="1" occurrence="1"
from="!HOME_NET" to="HOME_NET" port_from="ANY" port_to="ANY" plugin_id="1001" pl
ugin_sid="2020306" protocol="ANY">
          <rules>
                <rule type="detector" name="MC-SQLR Response" reliability="5" occur
rence="5" from="ANY" to="1:DST_IP" sticky_different="SRC_IP" time_out="600" port
_from="ANY" port_to="ANY" plugin_id="1001" plugin_sid="2020306" protocol="ANY">
                     <rules>
                          <rule type="detector" name="MC-SQLR Response" reliability="
8" occurrence="10000" from="ANY" to="1:DST_IP" sticky_different="SRC_IP" time_ou
t="86400" port_from="ANY" port_to="ANY" plugin_id="1001" plugin_sid="2020306" pr
otocol="ANY"/>
                     </rules>
                </rule>
          </rules>
     </rule>
</directive>
```

图 4-66　复杂的嵌套规则

下面是一个比较复杂的嵌套规则，如图 4-67 所示。

图 4-67　规则树的嵌套

对于 Userdata 各字段显示如图 4-68 所示。重要字段含义如下：

- Username：事件中指定的用户名。

- Pass：事件中指定的口令。
- Userdata1~Userdata8：事件中指定的数据域。

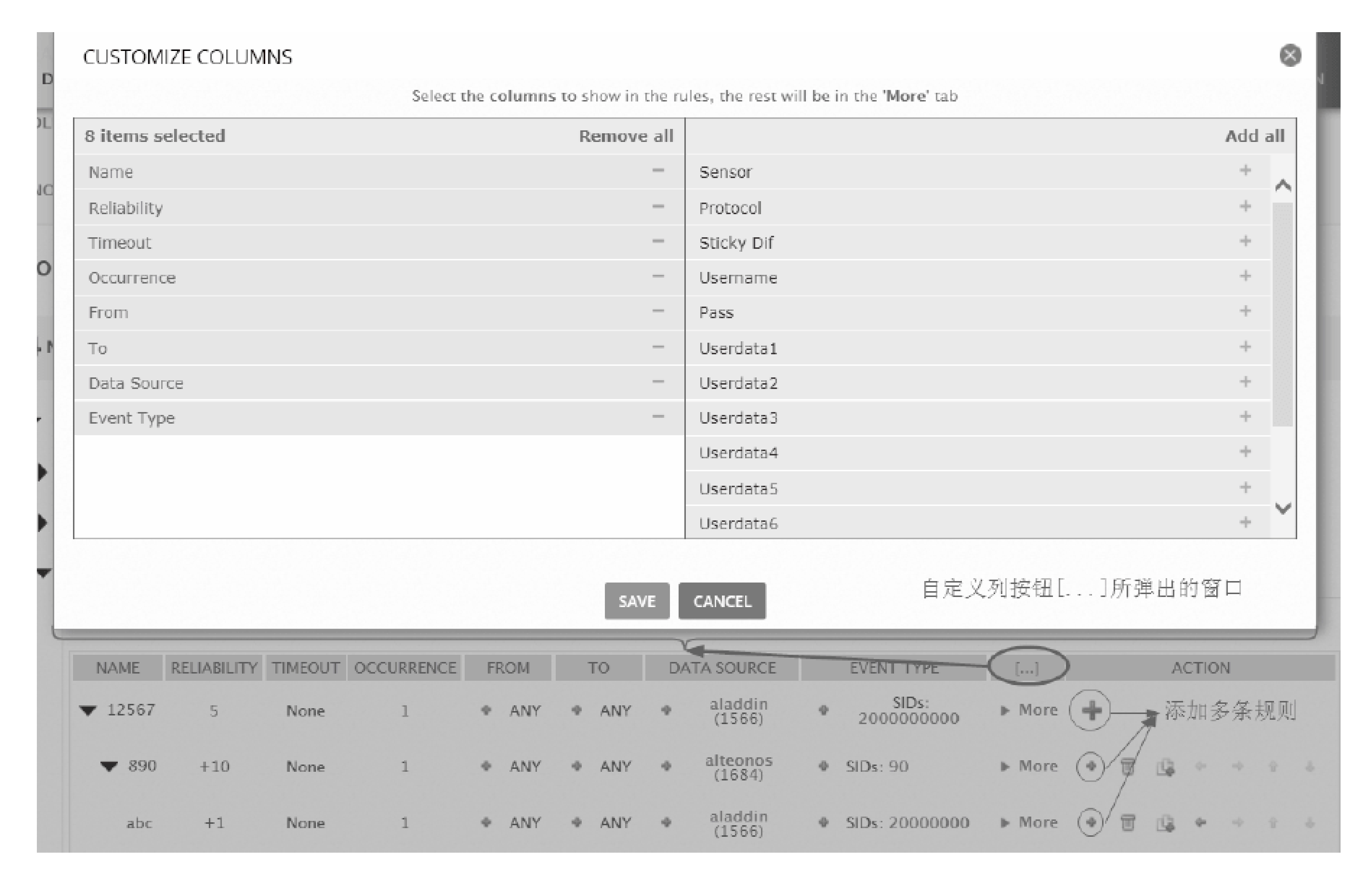


图 4-68　查看更多关联规则的属性

属性中列出 From、To、Data source 、Event Type 下方的 ✚ 号表示可修改，Action 下的 ✚ 号代表可添加规则 ，但系统默认指令中的规则不允许修改。

4.8.2　构建规则

SIEM 在攻击检测中发挥着重要作用，它能将收集的数据包括网络基础设施、网络应用、各种网络流量、漏洞扫描结果等信息，以图表的方式直观地呈现给用户。对于 SIEM 而言数据越多，越有利于分析结果的准确性。

建立 OSSIM 关联规则库是一个迭代过程，需要你结合企业自身情况进行长期的精细化调整，若没有规则，则无法分析数据。面对成千上万种可能建立的规则，我们自己要明确哪些规则是我们急需的，这就需要运维人员不仅仅是系统的管理者，有时候需要将自己放在攻击者的角色，并开始监视攻击者想“窃取”的数据或对系统进行模拟渗透，从这个层面开始制定防御检测规则，在编写规则时，将一些关键的攻击向量写入规则。接下来需要模拟一些攻击以完善规则。

开源版 OSSIM 系统中默认提供了 84 种关联检测规则，在 OSSIM USM 4.15 商业版本中提供了 2024 种规则，具体位置在菜单 Configuration→Threat Intelligence→Directives 中查看。

在学习关联规则中经常碰到各种指令，这些指令常用 XML 来描述，关联规则系统的规则属性，含义如下：

（1）Id：该属性允许定义相关联指令的唯一标识。这种编号必须遵循由 OSSIM 发布的指令。

（2）Name：此属性允许定义指令的名称（当指令匹配时显示）。

（3）SRC：源 IP 地址。

（4）DST：目标 IP 地址。

（5）Port_from：源端口。

（6）Port_to：目标端口。

（7）Priority：此属性允许定义关联指令的优先级。

（8）Type：该属性定义规则类型。仅有两种类型的规则。

（9）Detector：使用检测部件信息的规则，其包含于服务器数据库。

（10）Reliability：此参数越大（接近 10），表明报警越真实。此参数在关联过程中至关重要。实际上，随着规则的陆续匹配，本组报警误报的概率会降低。所以有可能在每个标记规则中修改高级报警的可靠性。其后的规则会以相对（例如：+3，意味着相对于前面的规则，全局可靠性提高了 3 个等级）或者绝对（例如：7，表明现在的可靠性等级是 7）的方式估计其等级。

（11）Occurence：出现频率。

（12）Plugin_id：此属性定义由规则预测的报警的来源。实际上，每个插件都有一个相关标识，此标识允许在相关性规则中引用该插件。

（13）Plugin_sid：此参数定义了和插件相关联的事件。通过在配置菜单的插件子菜单中，单击所需 plugin_id 便可以配置 plugin_sid。例如，由 plugin_id 1501 和 plugin_sid400 提供的报警等同于："apache：错误请求"，有了这两种属性（plugin_id 和 plugin_sid），就可以精确定义规则所预计的事件。

（14）Time_out：超时设定，此属性允许表明符合某个规则事件的等待时间。如果此事件没有在给定的时间（属性以秒计算）内发生，相关性指令会结束，并返回到前面规则计算的结果。

（15）Protocol（协议）：此属性允许配置由规则预计的网络事件类型。可以定义三种类型的协议：TCP、UDP、ICMP。这意味着有可能重新使用和上一个规则匹配的协议类型。所以，只需做如下表明：protocol= "1:PROTOCOL"，以此来明确表达此规则的协议和上一级规则相匹配的协议。如果要恢复二级规则匹配的协议，可表示为 protocol= "2:PROTOCOL"。

（16）From：此属性表明预报警的 IP 源地址。可以使用 6 种不同的方法：

- ANY，表明任意地址源和此属性匹配。
- x.x.x.x 指定 IP 地址。
- 通过引用和引用协议属性原则一致。例如，1：SCR_IP= 和一级相关指令匹配的报警的源地址，2：DST_IP=和二级相关性指令报警的目标地址。这样的规则参看图 4-69 所示的例子。

图 4-69 引用协议

- 网络名称，可以通过在框架下定义的网络名称来使用通过某个范围的 IP 地址。变量 HOME_NET 定义了与其相关的所有在框架下定义的网络。
- 特定地址，表示由破折号分隔的几个 IP 地址。
- 拒绝，此允许拒绝部分 IP 地址或者网络名称，例如：!192.168.2.3，HOME_NET。

4.9 深入关联规则

4.9.1 基本操作

OSSIM 中的告警关联组件，利用预先定义的规则（/etc/ossim/server/*.xml）进行关联分析。用户将如何对关联规则进行操作呢？我们通过 Web 界面，新建一个 Correlation Directives，新建两个规则 ssh 和 test，然后我们查看详细内容，路径在/etc/ossim/server 目录，名为 user.xml 文件。其他默认规则由/etc/ossim/server/directives.xml 定义，如图 4-70 所示。

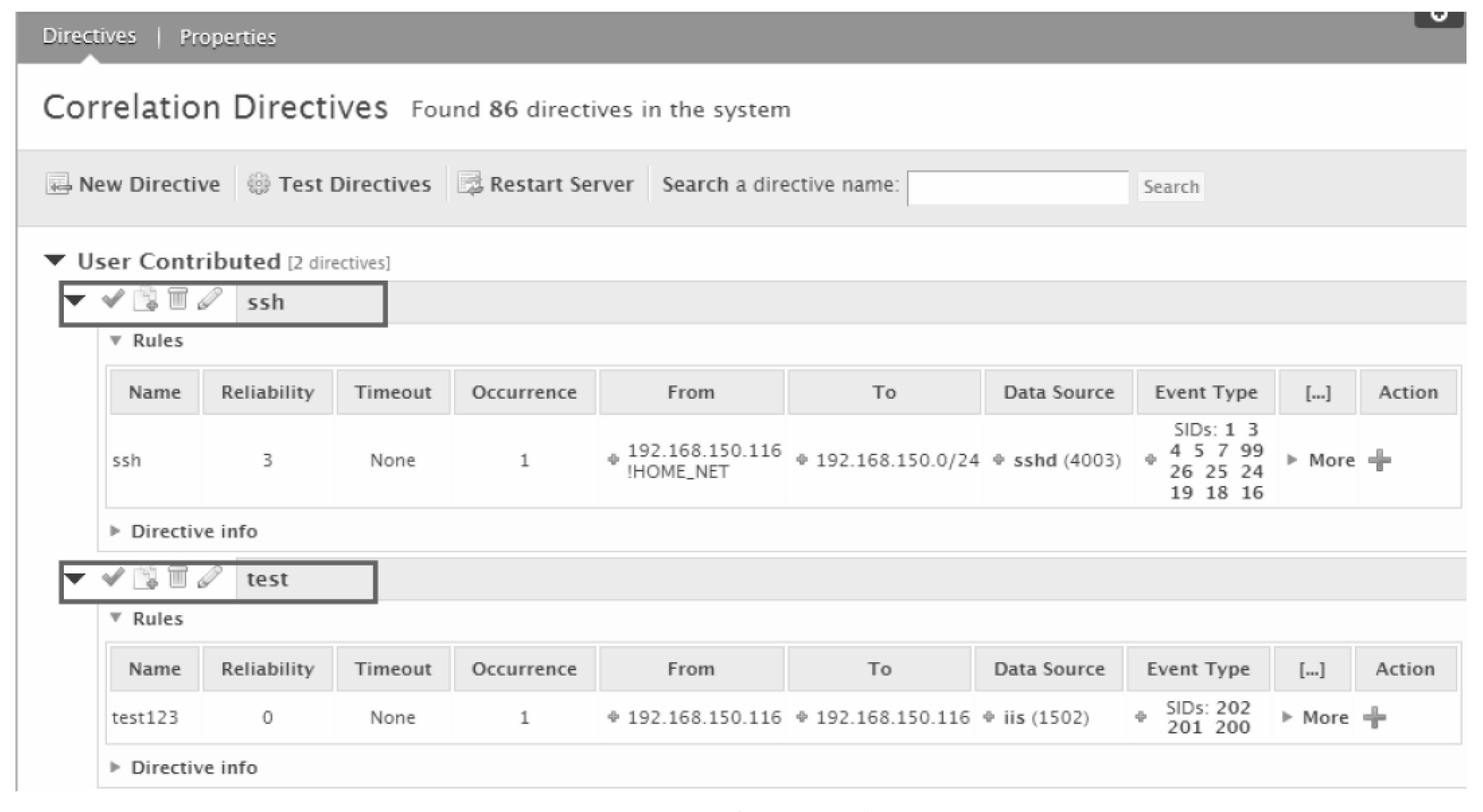


图 4-70 自定义指令

当系统调用 Directive 下的策略时，根据 categories.xml 配置文件读取相应的 XML 配置文件，这些策略文件存储位置为/etc/ossim/server/，文件功能如下:

- alienvault-attacks.xml：AlienVault 攻击类策略。
- alienvault-bruteforce.xml：AlienVault 暴力破解类攻击。
- alienvault-dos.xml：Alienvault 拒绝服务类。
- alienvault-malware.xml：Alienvault 恶意软件（包括检测各种蠕虫的规则）。
- alienvault-misc.xml：Alienvault 各种失误类（Miscellaneous）。
- alienvault-network.xml：Alienvault 网络类（开源版没内容）。
- alienvault-policy.xml：Alienvault 策略。
- alienvault-scan.xml：Alienvault 扫描。
- user.xml：Alienvault 用户自定义。

如果 OSSIM 关联引擎读不到这些策略配置文件，则在 Analysis→Alarm 中不会生成报警。接下来开始解读 XML 规则实例。

4.9.2　理解规则树

关联引擎通过规则来实现对安全事件的关联分析，读者需要看懂 OSSIM 的规则，其中采用了特有的树形规则，在规则树中的每一个节点对应一条关联规则（rules）。在匹配时关联引擎将某段时间内收到的统一格式的报警，从根节点开始往叶子节点逐次匹配，系统根据匹配的结果，可以进行事件聚合和再次提升报警级别。

从图 4-71 中可以看出，这种基于事件序列的关联方法，每条指令相当于一颗由规则组成的树，所以可以把它叫做基于树形指令（Directive）的关联方法，基本思想是根据相关事件序列，创建一系列的规则来描述攻击场景。

```
<?xml version="1.0" encoding="UTF-8"?>

<directive id="500001" name="ssh" priority="3">
   <rule type="detector" name="ssh" from="192.168.150.116,!HOME_NET" to="192.168.150.0/24" port_from
="ANY" port_to="ANY" reliability="3" occurrence="1" plugin_id="4003" plugin_sid="1,3,4,5,7,99,26,25,
24,19,18,16" username="admin" password="MTIzNDU2Nw=="/>
</directive>

<directive id="500002" name="test" priority="3">
   <rule type="detector" name="test123" from="192.168.150.116" to="192.168.150.116" port_from="ANY"
port_to="ANY" from_rep="true" to_rep="true" from_rep_min_pri="5" to_rep_min_pri="5" from_rep_min_rel
="5" to_rep_min_rel="5" reliability="0" occurrence="1" plugin_id="1502" plugin_sid="202,201,200"/>
</directive>
```

图 4-71　自定义指令内容

在 OSSIM 系统中由 XML 语言定义关联序列，关联分析引擎启动时，将所有关联导入每个关联规则序列，并由下面标签组成，将实际 XML 提炼后，得到如下模板：

```
    <directive id="" name="" priority="">
    <rule type="" name="" reliability="" occurence="" from="" to="" port_from=""
port_to="" plugin_id=""plugin_sid="">
    <rules>
      <rules>
```

```
    ......
</rules
     ......
</rule>
</directive>
```

每个序列起始包括两个标签“directive_id”和“directive name”，嵌套中的每个序列又包括多个规则，规则之间又可嵌套规则，即可以递归。

其中 Rule 表示规则，规则后面描述了一个可能发生的攻击场景，Event 代表在当前已发生的攻击场景下，一个攻击场景由若干条规则组成，这些规则可能是“与”和“或”的关系。这样表示的好处是如果发生一个攻击场景，那么只需要满足哪些规则即可。

通过计算每个 Rule 里的 Reliability 值，以确认该攻击发生的可信度为多少。其中场景（Scene）id 用 directive_id 来表示。前面讲过 Reliability 可以是 0~10 的整数，直接给可靠性赋值，也可以使用一个变量，表示当这个规则满足时，将上一步攻击场景的可靠性做修改，将结果存入 New_Reliability 中。规则部分的其他属性代表了将它做匹配的 Event 的属性值，而与数据库的具体交互是通过 Action 来表示。

我们看个实际的例子，下面这段指令主要用来检测公网服务器是否可用，其中包含了两个规则。

```
    <directive id="500000" name="Public Web Server unavailable" priority="1">
        <rule type="detector" name="server unavailable" reliability="1"
       occurrence="1" from="192.168.11.100" to="ANY" port_from="ANY"
port_to="ANY"
       plugin_id="1525" plugin_sid="1,7,9">
       <rules>
       <rule type="monitor" name="Ping Baidu Server  in order to check internet
connection"
         reliability="10" from="www.baidu.com" to="www.baidu.com"
plugin_id="2009" plugin_sid="1" condition="gt" value="0" interval="20"
time_out="120" />
       </rules>
    </directive>
```

关键字段解释：

（1）Detector：检测器规则，自动收集从代理发来的记录，包括 Snort、Apache、Arpwatch 等。

（2）Monitor：监控器负责查询 Ntop 服务发来的数据和会话。

（3）Name：事件数据库中的规则名称。

（4）Reliability：可靠性。

（5）Plugin_id：插件 ID，查看更多插件 ID 请参考第 1 章的表 1-7。

（6）Plugin_sid: 插件子 ID 号，分配给每个插件事件的子 ID，比如 Snort 这个插件 ID 号为 1001，而 1501 就是它的子 ID 号，代表 Apache 事件的响应代码，当然不止这一个。

（7）Condition：条件参数和下面 6 个逻辑有关系，必须符合的逻辑条件匹配规则如下:

- eq：等于（Equal）。
- ne：不等于（Not equal）。
- lt：小于（Lesser than）。
- gt：大于（Greater than）。
- le：小于等于（Lesser or equal）。
- ge：大于等于（Greater or equal）。

（8）Interval：间隔，这个值类似于 time_out，用于监控类规则。

又如 OSSIM 的例子，我们查看/etc/ossim/server/alienvault-scan.xml 这个扫描攻击场景的策略文件，描述的结构就像逻辑树一样，基本格式如下：

```
  Gree:level 1
     Yellow : level2
      Orange:level3
      Red: level 4
```

下面给出部分代码内容，如图 4-72 所示。

```
<directive id="30003" name="AV-FREE-FEED Network scan, SIP service discovery activity from SRC_IP" p
riority="4">
   <rule type="detector" name="SIP service discovery activity detected" reliability="2" occurrence="
1" from="!HOME_NET" to="HOME_NET" port_from="ANY" port_to="5060" plugin_id="1001" plugin_sid="200860
9,2008641" protocol="UDP">
        <rules>
             <rule type="detector" name="SIP service discovery activity detected" reliability="4" oc
currence="4" from="1:SRC_IP" to="1:DST_IP" sticky_different="DST_IP" time_out="360" port_from="ANY"
port_to="5060" plugin_id="1001" plugin_sid="2008609,2008641">
                 <rules>
                     <rule type="detector" name="SIP service discovery activity detected" reliabilit
y="8" occurrence="10000" from="1:SRC_IP" to="1:DST_IP" sticky_different="DST_IP" time_out="360" port
_from="ANY" port_to="5060" plugin_id="1001" plugin_sid="2008609,2008641"/>
                 </rules>
             </rule>
        </rules>
    </rule>
</directive>
```

图 4-72　攻击扫描指令示例

（9）Occurrence，表示发生次数，默认为 1，攻击场景不同，这个值也不一样。这个值越大，越要引起管理员的注意。这里表示的发生次数，也就是计算具有相同的“from、to、port_from、port_to、plugin_id、plugin_sid”发生次数，以便代入到关联模式中的下一个规则中。

这个数值在基于规则的关联分析中非常有用，例如当目标 IP 地址发生的异常行为，事件数量发生的频率达到了一定数值时（某人针对该系统发动攻击），OSSIM 会发出预警提示，管理员就可以有针对性地开展深入调查。当然对源 IP 也适用。

From 表示来源，源地址有以下几种形式：

- ANY，任意 IP 地址都可以匹配。
- 小数点和数字的 IPv4 形式。
- 以逗号隔开的 IPv4 地址，不带掩码。
- 可以使用任意数目的 IP 地址，中间用逗号隔开。

- 网络名称，可以使用网络中事先定义好的网络名称。
- 相对值，这种情况比较复杂，可以引用上条规则中的 IP 地址，例如:
  - 1:SRC_IP 表示引用前一条规则的源地址。
  - 2:DST_IP 表示引用前第二条的目的地址作为源地址。
- 否定形式，可以使用地址的否定形式如："!192.168.150.200，HOME_NET"。

如果 HOME_NET = 192.168.150.0/24，将匹配一个 C 类子网排除 192.168.150.200。

（10）Time_out：表示超时，其等待一定时间以匹配规则，时间超出则匹配失败。

（11）Port_from/Port_to：表示来源/目的端口，Port_from 可以是 ANY，Port_to 可以是一个端口号或一个逗号分隔的端口序列，1:DST_PORT，也可以否定端口，例如，port="!22,25"。注意：dst_ip 表示目的 ip 地址，而 dst_port 则表示目的端口号，ipaddr 本地 ip 地址。

（12）Protocol（协议）：可以使用以下字符串：TCP、UDP、ANY。

在 OSSIM 系统运行前，必须为已知的攻击场景建立对应的树形规则集，在启动时，系统将预先定义好的指令读入内存，在接收到一个事件（event）后，先将该事件与之前已经匹配、而还没有匹配完的指令中的规则进行匹配，然后再与其他规则匹配。那么在一个复杂的攻击场景中，一条 Event 就会与多条规则匹配，如果此事件的风险值超过预先设定的阈值，则将其 alarm 属性设为真，并在 Alarm 界面中显示。

例如：plugin_id="1001" plugin_sid="2008609,2008641"。

在 OSSIM 4.6 系统中，有 385 个数据源，这里 ID=1001，代表 Snort 检测插件，产品类型属于 IDS，主要适用于 Snort 规则，其他插件 ID 还记得吗？如果忘了请返回本书第 1 章，查看“插件&功能”对应表。

实际上在 OSSIM 系统中，Snort 插件 ID 范围是 1001～2145。在 OSSIM 4.6 版中，Configuration→Threat Intelligence→Data Source 可以查看到所有数据源。如图 4-73 所示。

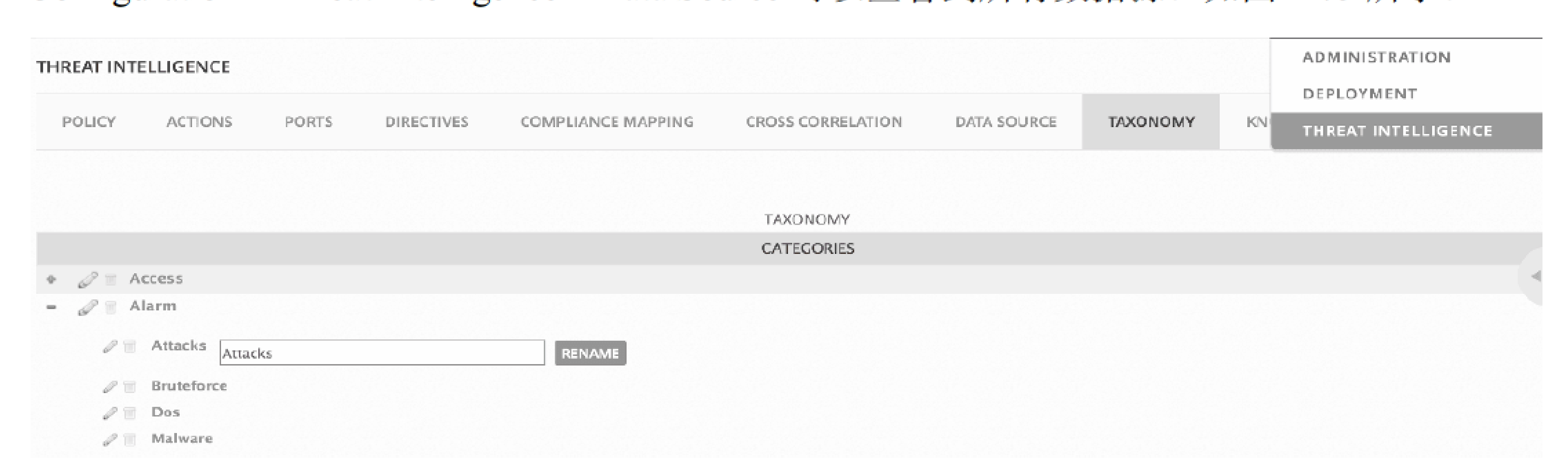


图 4-73 数据源分类

在 OSSIM 的关联引擎运行之前，必须为所有已知的攻击场景建立对应的树形指令，这也是它的核心价值的体现。在 OSSIM 启动时，系统会将所有预先定义好的指令读入内存，在接收到一个 directive_event 后，先将该事件指令中的规则匹配，然后再与其他指令的规则匹配，注意一个事件可以与很多指令中的规则匹配。

4.9.3　攻击场景构建

黑客攻击时往往带有目的性，不会随意扫描，而有目的的攻击总有个顺序，可以用一组攻击行为来描述。下面是针对 DNS 服务器实时 WinNuke 攻击的例子，WinNuke 攻击又称带外传输攻击（一种拒绝服务攻击），被攻击的目标端口通常是 139、138、137。其过程如图 4-74 所示。

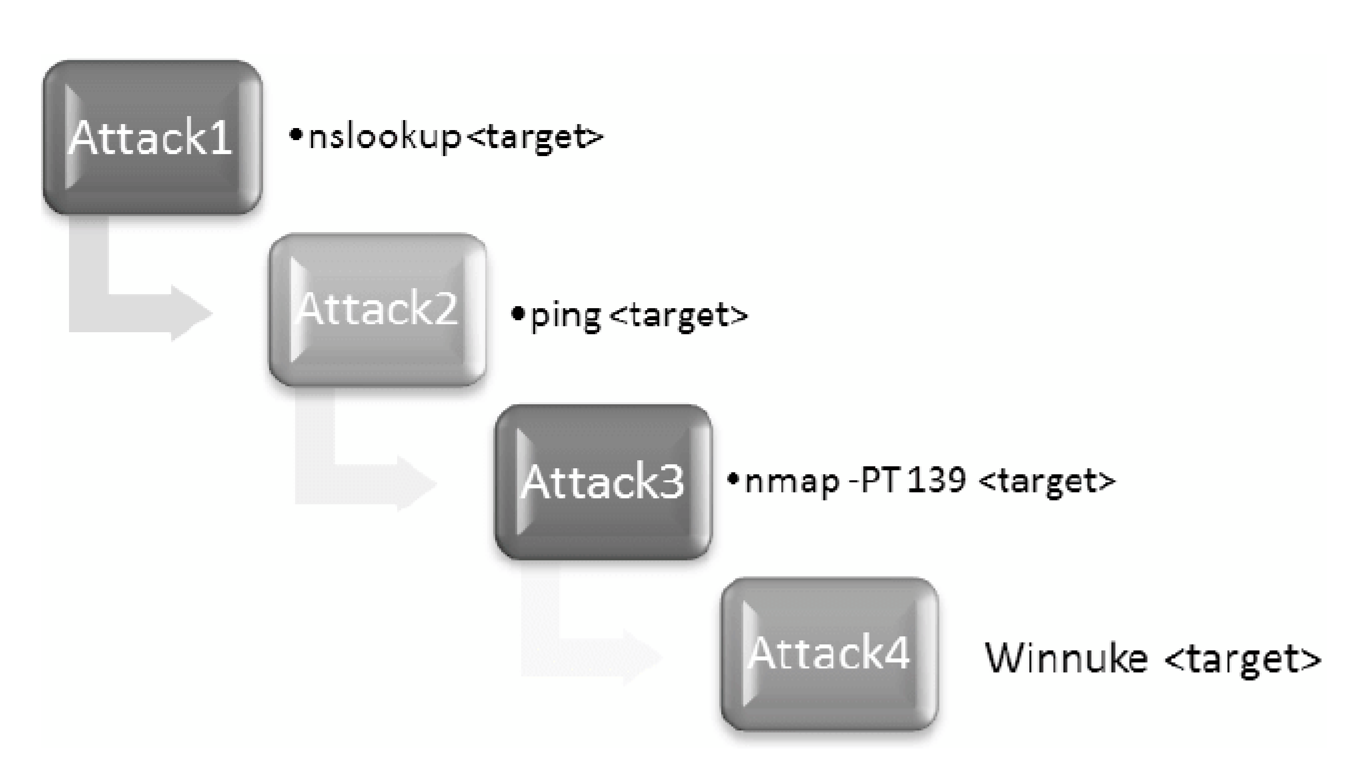


图 4-74　Winnuke 攻击场景

这个攻击包含相互关联的 4 个步骤，细节大家可以参考 NASL 脚本 winnuke.nasl，检查 DNS 服务器是否受到了攻击影响。

另外，我们可以使用一个攻击场景测试数据集 LLDOS1.0，其中包含了一个复合攻击过程，大致攻击序列分为 5 个阶段：

（1）通过 IP sweep 进行活动主机探测；

（2）使用 Sadmind Ping 进行 sadmind daemon 服务端口扫描，探测可能存在的 sadmind 漏洞主机（例如 Solaris 8）；

（3）利用主机上的 sadmind 漏洞进行系统入侵，获得 N 台主机的 root 权限；

（4）在攻破的主机上安装用户 DDoS 攻击的木马；

（5）利用被控主机对目标发起 DDoS 攻击。

以上攻击场景可描述为：

```
IP_Sweep→RPC_port_scan→sadmin_overflow→Remote_login→DDoS
```

有关这种攻击防范案例，读者可以参考《Unix/Linux 网络日志分析与流量监控》一书 7.2 节内容。

4.9.4　报警聚合计算方法

下面我们分析通过 IP 来聚合一类安全告警，这里用到源地址或者目标地址的相似度计算来实现报警聚合。

（1）IP 聚合的计算

本例中源地址和目标地址的 IP 地址用十进制和二进制表示，采用下面的方法计算相似度。

```
IP1： 192.168.1.9   11000000.10101000.00000001.00001001
IP2： 192.168.2.3   11000000.10101000.00000010.00000011
Mask 255.255.255.0 11111111.11111111.11111100.00000000
```

IP1 和 IP2 之间的距离用 Mask 掩码中 1 的位数表示，故它们的距离为 22，如果 Mask 为 32 说明两个 IP 完全相同，如果 Mask 为 0 说明两个 IP 绝对不可能是同一子网的 IP，通过公式可以将这种相似度量化为 0~1 的小数。

量化后我们就可以写规则了，例如:

```
Rule {Similarity[1]SRC=1：SRC_IP
DST=1：DST_IP
Similarity[2]={1,0.75}
Occurrence=100
… …}
```

SRC 和 Similarity[1]表示要求与这个节点匹配的报警的源 IP 与该指令根节点的源 IP 相同，DST 和 Similarity[2]=0.75，表示目标 IP 和根节点目标 IP 距离大于 24，即 Mask=255.255.255.0，Occurrence=100 表示关联引擎至少收到 100 条以上的报警才匹配。

（2）端口扫描攻击事件告警的聚合计算

这种方法和 IP 相似度计算相同原理，对于攻击中使用到端口也可以采用这种方法，具体规则不再举例。

4.10 自定义策略实现 SSH 登录失败告警

为防止用户反复登录暴力破解密码，我们一方面需要加固 SSH 服务，另一方面还需要监控 SSH 登录情况，对于非法登录一定要记录在案，一旦 SSH 登录失败的次数超过事先约定的阈值，系统便报警。要实现这一目的，可利用 OSSIM 关联规则，观察一段时间内发生的事件，如果超过阈值，警报就会触发。例如关联规则匹配“在 5 秒内登录失败 5 次的事件”，这会被显示为报警。如果在 5 秒内登录失败次数达 40 次呢？这也会被归为同一类报警，而不会重复多次。

关联规则与病毒特征库不同，OSSIM 内置的规则提供了基本的分析思路，大家在日常使用中切不可生搬硬套，要根据自身环境调整，在理解已有的规则后，根据所包含的攻击场景，推理条件来编写新的规则。当然规则并不是一成不变，也要随着业务的变化而相应改变规则。

接下来，介绍 OSSIM 4 系统下的一个实用功能，就是可以为网络的 SSH 服务器提供多次失败自动报警功能，实现方法是在 Configuration→ThreatIntelligence→Directives 菜单下选择“New directive”按钮，可以设置 SSH 登录失败报警策略，例如图 4-27 中，新建一条策略名称为“ssh_attack”，设定登录失败就视为风险存在，然后通过最大尝试次数为 3 次，指定报警。详细步骤如图 4-75 ~图 4-80 所示。

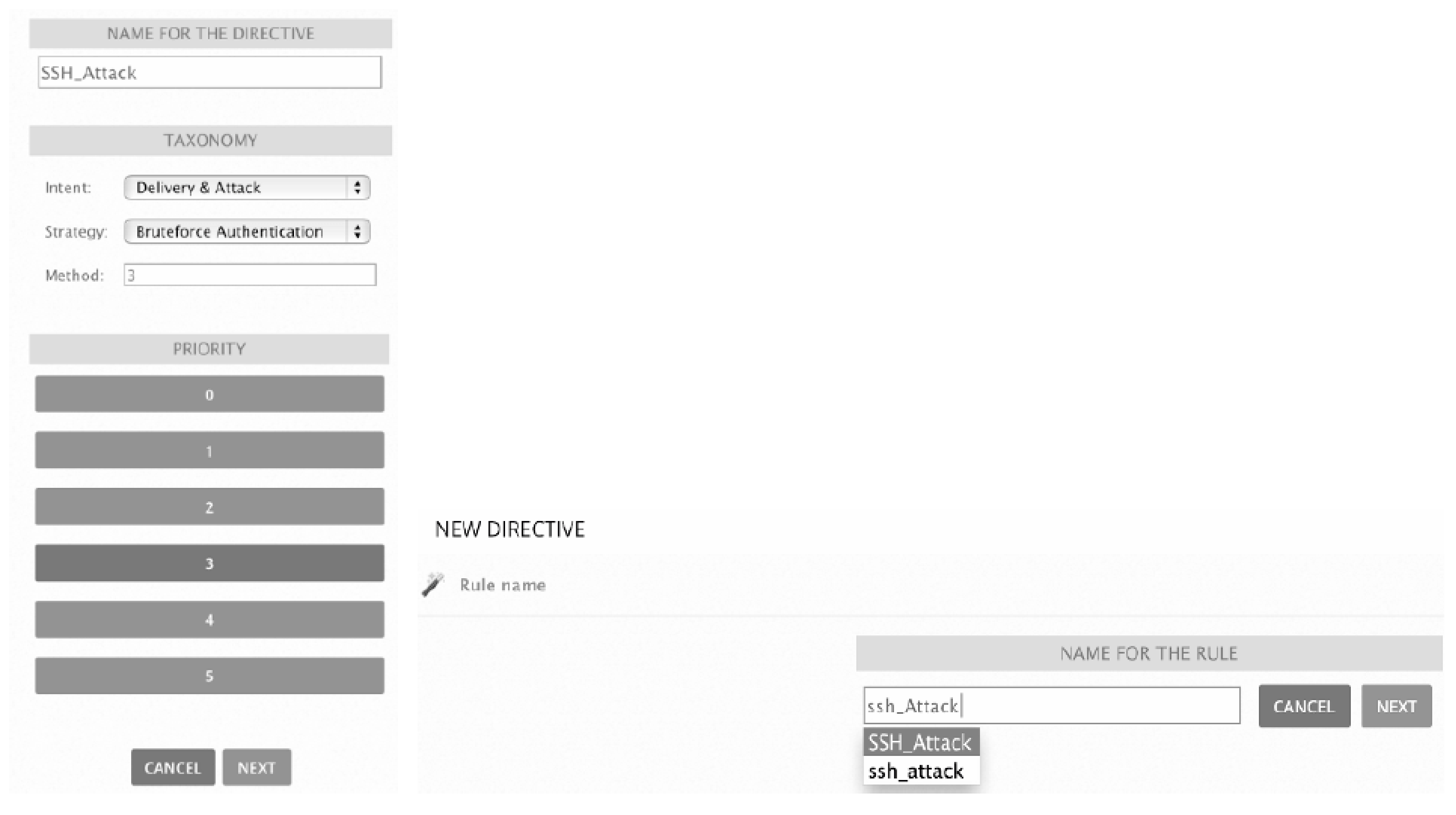


图 4-75　添加指令、输入规则名称

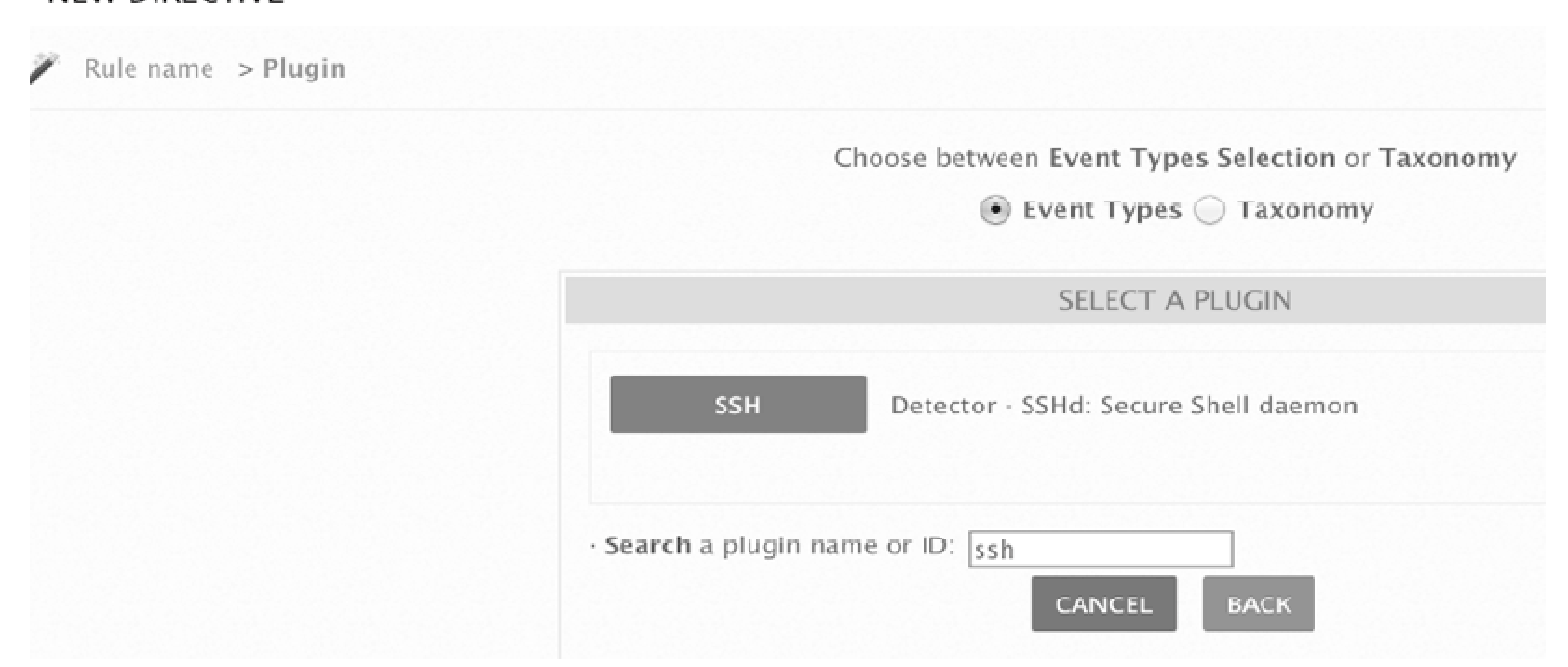


图 4-76　选择事件类型

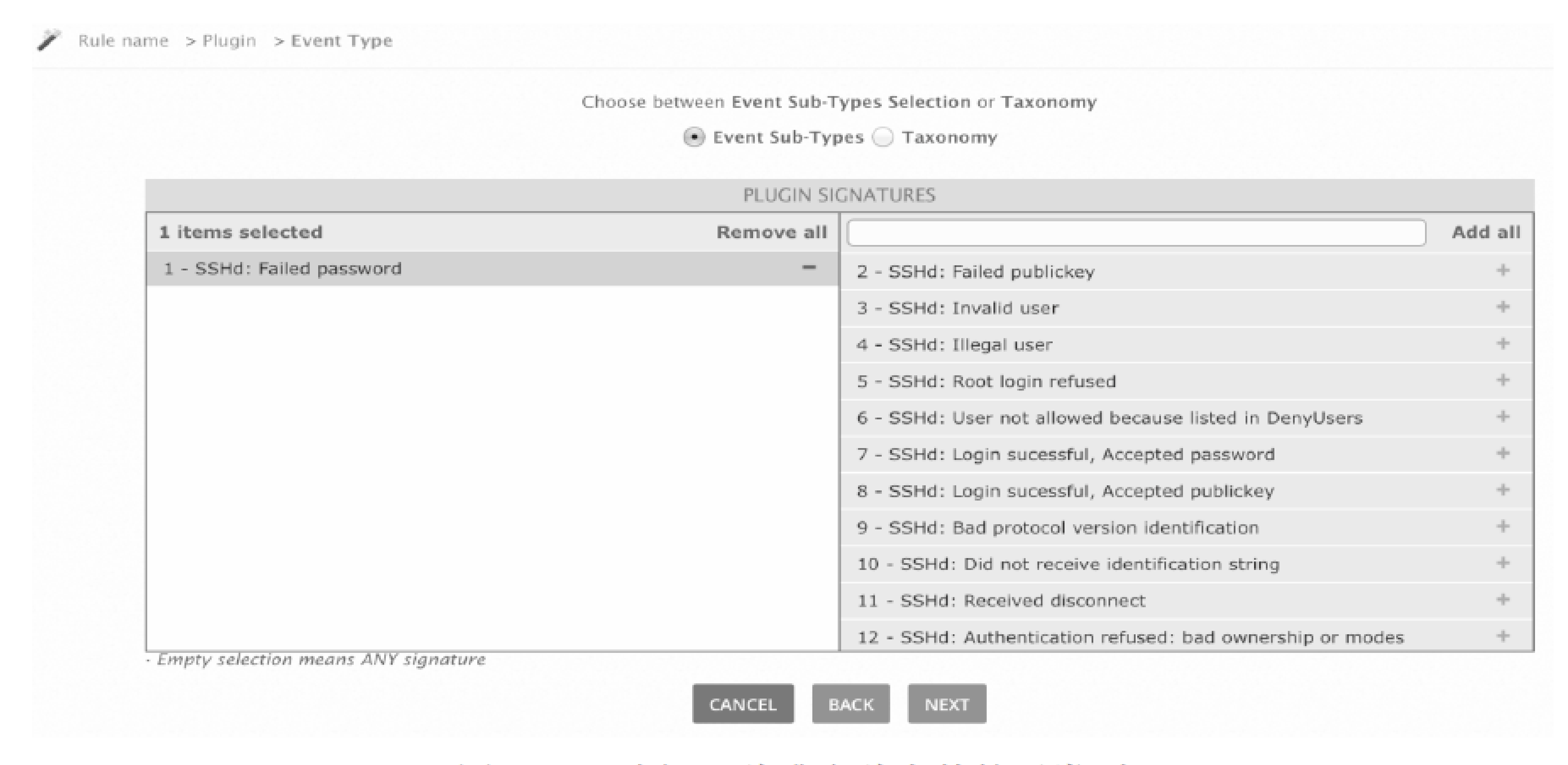


图 4-77　选择一种或多种事件的子类型

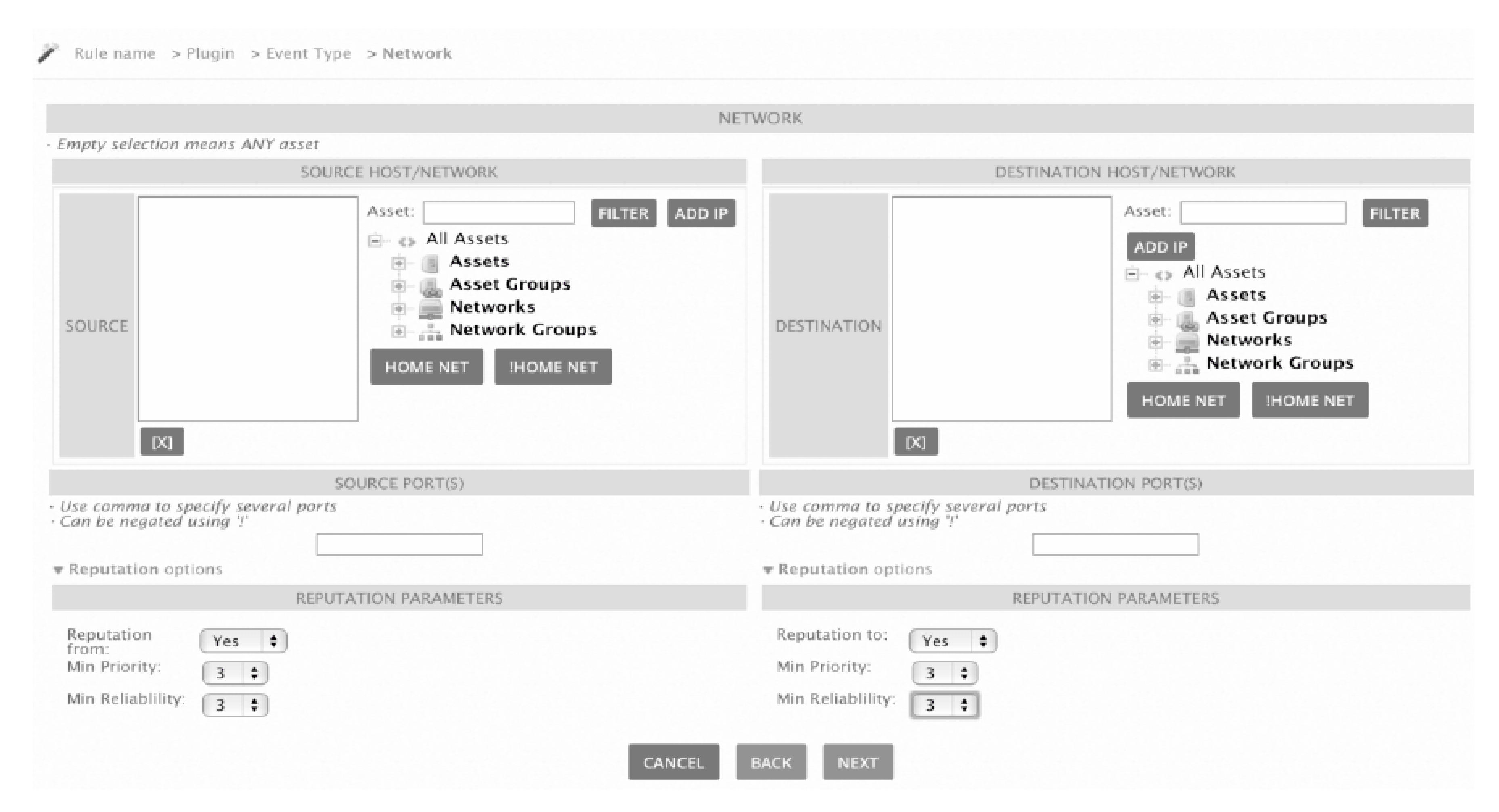


图 4-78　选择源和目标地址

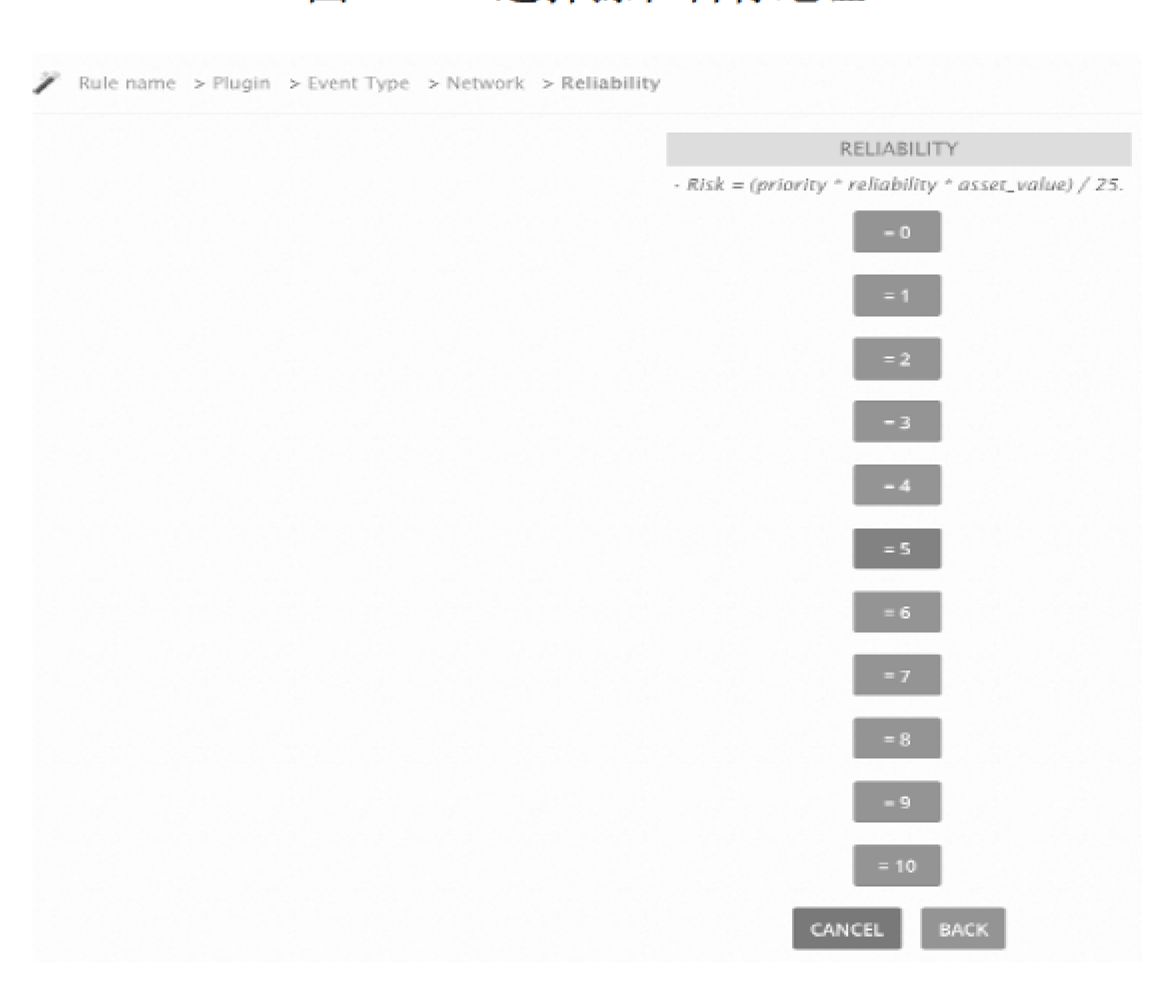


图 4-79　设定策略可靠性

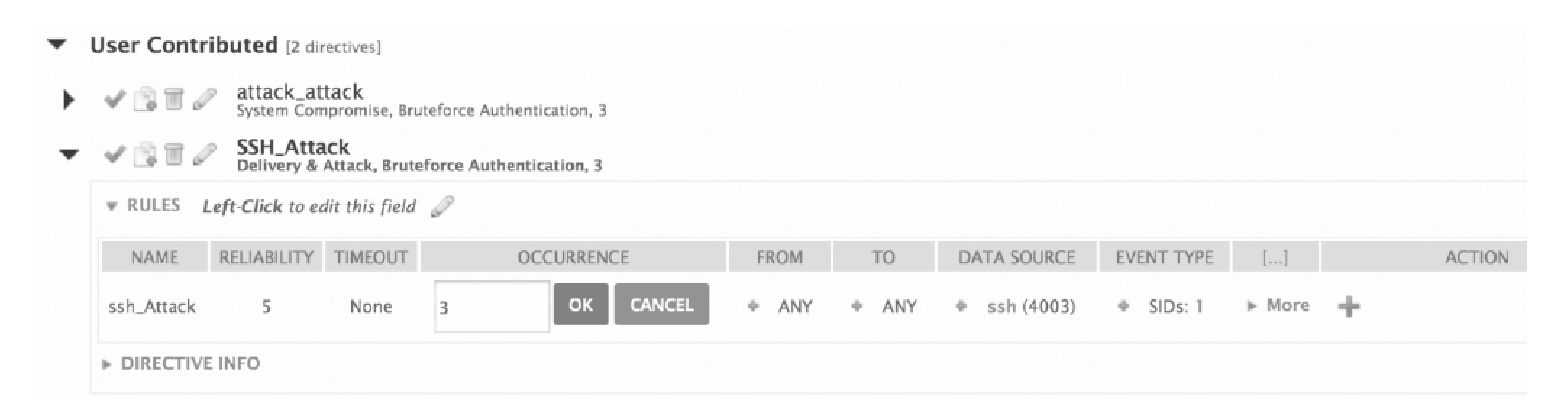


图 4-80　设定发生此类事件的次数

设置好以后，单击确定按钮，我们可以在列表中查看到如下策略信息。如图 4-81 所示。这里 Directive 代表指令含义，实际上可以理解为添加一条用于检测 SSH 攻击的指令。

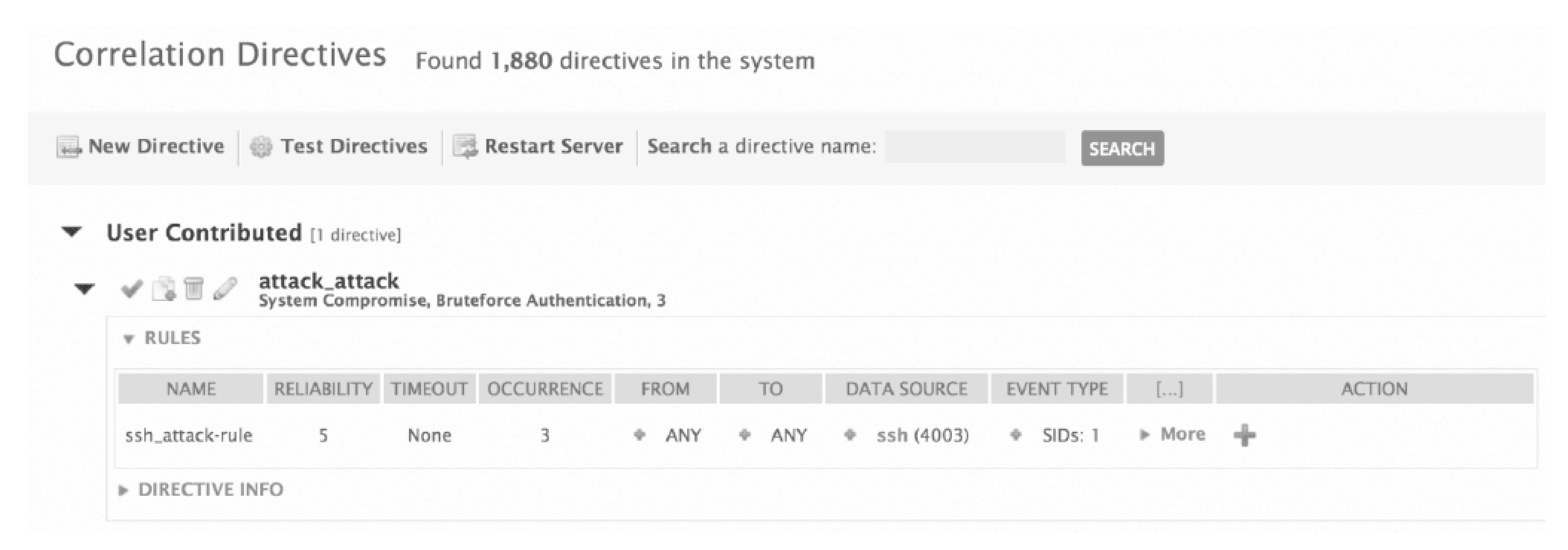


图 4-81　设置 SSH 攻击检测条件

当策略设置完毕，一定要单击“Restart Server”按钮，使设置生效。图 4-82 所示内容就是 SSH 客户端登录失败后在 SIEM 控制台上的提示。

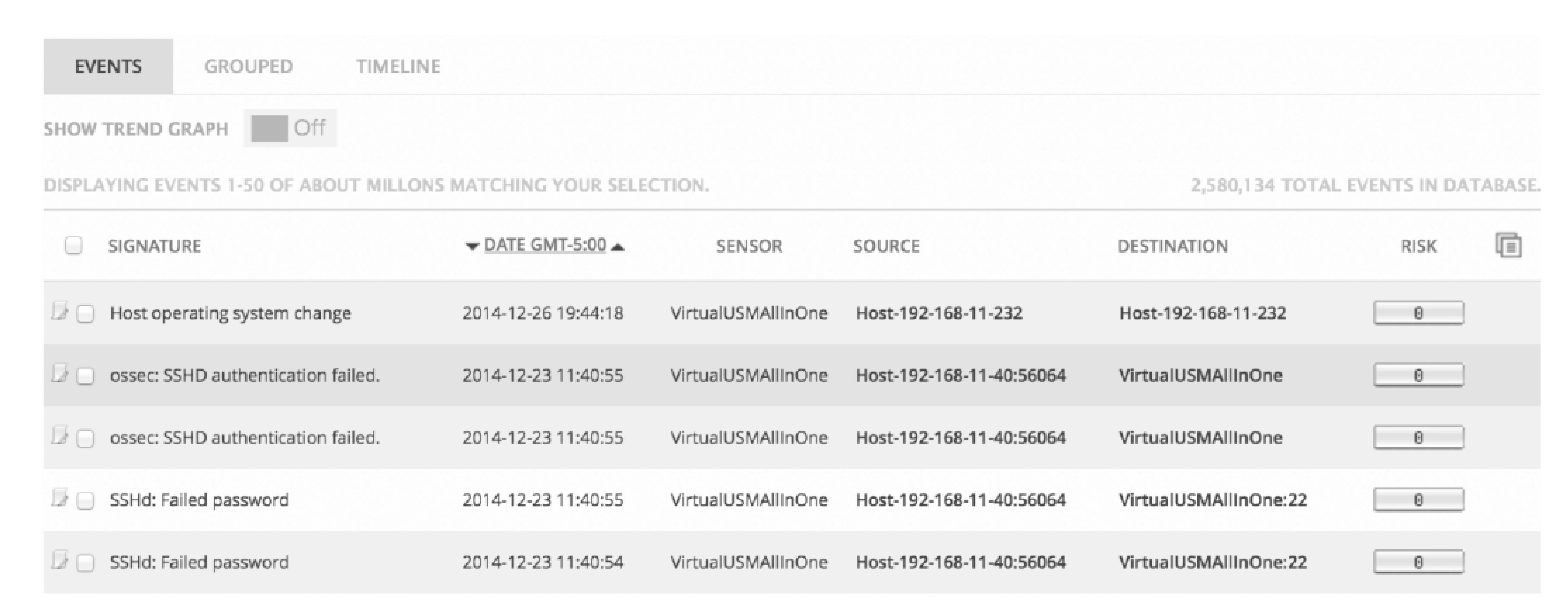


图 4-82　SIEM 检测到 SSH 登录失败记录

最后，使用 OSSIM 的日志筛选功能就知道所有的登录失败告警有多少。这对于掌握服务器的安全情况非常有帮助。但并不是所有报警信息都代表有攻击行为，其中有不少属于误报，还需要经验来综合判断。

4.11 小结

日志关联引擎是 OSSIM 日志和事件分析的核心。本章从关联分析入门开始讲起，逐步介绍了多种关联分析技术，最后介绍了 OSSIM 的通用关联检测规则，运用实例给读者展示了关联引擎的基本使用方法。

第 5 章

OSSIM系统监测工具

从本章节可以学习到：

- Linux 性能评估
- 常用监测工具
- OSSIM 压力测试
- 性能检测方法
- 性能分析工具
- 影响 MySQL 性能的因素

OSSIM 在实验室测试过程中，往往数据量小、数据种类单一。我们往往只在意功能的实现，而很难关注到性能的薄弱之处，等系统上线，实际运行一段时间后，才发现系统的性能出现问题，这时再来考虑提高系统性能则要花费更多的精力，所以需在系统投入生产之前做好各项测试和准备工作。对于系统评估和监控的常用工具和方法，读者需要掌握。尤其是 OSSIM 系统在高并发、大数据量的访问情况下，我们的系统会不会出现状况。

5.1 Linux 性能评估

当我们搭建好系统后，由于硬件、软件或是网络环境等问题都会对应用产生影响。作为系统工程师，就是要适应各种环境因素的变化来评估应用系统可能出现的各种异常情况，并定位故障、优化系统以解决这些比较难解决的问题。性能监测是系统优化过程中的重要一环，如果不清楚性能瓶颈在哪里，怎么优化呢？所以找到性能瓶颈是性能监测的目的，也是系统优化的关键。

5.1.1 性能评估工具

系统运行良好的时候恰恰也是各项资源达到了一个平衡，任何一项资源的过度使用都会造成平衡体系的破坏，从而造成系统负载极高或者响应迟缓，如 CPU 过度使用、系统响应变慢等。内存耗尽又会造成虚拟内存使用，使用虚拟内存又会造成磁盘 I/O 增加和 CPU 开销增加，故优化、监测、测试通常是连在一起的一个循环，而且是长期的过程，通常监测的子系统有以下这些：

- CPU
- Memory

- I/O
- Network

这些子系统互相依赖，了解这些子系统的特性，监测这些子系统的性能参数，及时发现可能会出现的瓶颈对系统优化很有帮助。综合运用常见的性能监控工具，对我们维护系统非常有帮助，下面通过 ps、head、awk 命令，按顺序从高到低显示 CPU 内存占用情况，如图 5-1 所示。

```
alienvault:~# ps aux |awk '{print $2,$3,$4,$11}' |head -1 && ps aux |awk '{print
 $2,$3,$4,$11}' |sort -k3 -nr |head -6
PID %CPU %MEM COMMAND
3253 1.3 9.0 /usr/bin/suricata
2905 0.6 5.7 /usr/bin/ossim-server
2504 0.4 3.2 /usr/sbin/mysqld
3654 0.0 3.1 /usr/share/alienvault/api_core/bin/python
3653 0.0 3.1 /usr/share/alienvault/api_core/bin/python
3652 0.0 3.1 /usr/share/alienvault/api_core/bin/python
alienvault:~# _
```

图 5-1　用命令行工具显示 CPU 内存情况

监控这些子系统有一些更专业工具，每个工具都侧重某一个领域，所以读者需要善于将多种工具配合使用，以发挥更大作用。常用的监测工具如表 5-1 所示。

表 5-1　常用监测工具

工具	简单介绍
top	查看进程活动状态以及一些系统状况，启动 top，按 Shift+N，可根据内存 CPU、PID、SWAP 等参数进行实时排序
vmstat	查看系统状态、硬件和系统信息等
iostat	查看 CPU 负载、硬盘状况
sar	查看系统状况
mpstat	查看多处理器状况
netstat	查看网络状况
iptraf	实时网络状况监测
tcpdump	抓取网络数据包，详细分析
mpstat	查看多处理器状况
tcptrace	数据包分析工具
netperf	网络带宽工具
dstat	综合工具，结合了 vmstat、iostat、ifstat、netstat 几个命令的功能
slabtop	显示内核片缓存信息

本章将介绍 OSSIM 中各种常用工具，待大家熟练掌握之后就可以灵活运用。性能调优对新手而言难度较大，不同的系统，其用途也不同，要找到性能瓶颈，需要知道系统有什么应用、有什么特点，如网站服务器对系统的要求肯定和文件服务器不一样，所以区分不同系统的应用类型很重要，通常应用可以分为两种类型：

- I/O 相关，I/O 相关的应用通常用来处理大量数据，需要大量内存和存储，频繁 I/O 操作读写数据，而对 CPU 的要求则较少，大部分时候 CPU 都在等待硬盘，如数据库服务器、文件服务器等。

- CPU 相关，CPU 相关的应用需要使用大量 CPU，如高并发的 Web/Mail 服务器、数据包抓包分析等都可被视作 CPU 相关的应用。

5.1.2 查找消耗资源的进程

在大负载服务器中，我们经常需要查明最消耗资源的进程，可通过“top -c”、“pstree -Aup”、“ps -ef”查出，如可疑进程 PID 为 41002，用下面命令找到对应的进程名：

```
#ps -p 41002 -o cmd
```

如果“ps -ef”也无法查出问题，还可用 strace 命令:

```
例如：#strace -o strace_psef.log -f ps -ef
```

5.2 OSSIM 压力测试

前几章讲到 OSSIM 系统由若干开源安全系统所组成，对于这样一个复杂系统，部署完毕后，系统到底怎么样、稳定性如何等一系列问题，我们需要经过一些测试才能知晓。通常，对防火墙、入侵检测的测试和评估有着严格的测试方法和流程，下文仅对 OSSIM 系统中常见的日志流量和网络数据包流量进行仿真，另外还包括 MySQL 的压力测试。

5.2.1 软硬件测试环境

- 千兆交换机一台，交换机必须具有端口 SPAN 功能，本次测试中 OSSIM 服务器与 SPAN 口连接，使得它能够监听到其他端口的测试数据流。
- 装有 OSSIM 4.3 系统的测试服务器一台。
- 高性能 PC 机 2 台，分别模拟两台客户端，操作系统 Linux、Windows 均可。这里选用 BT5（Back Track，以后简称 BT），拓扑如图 5-2 所示。

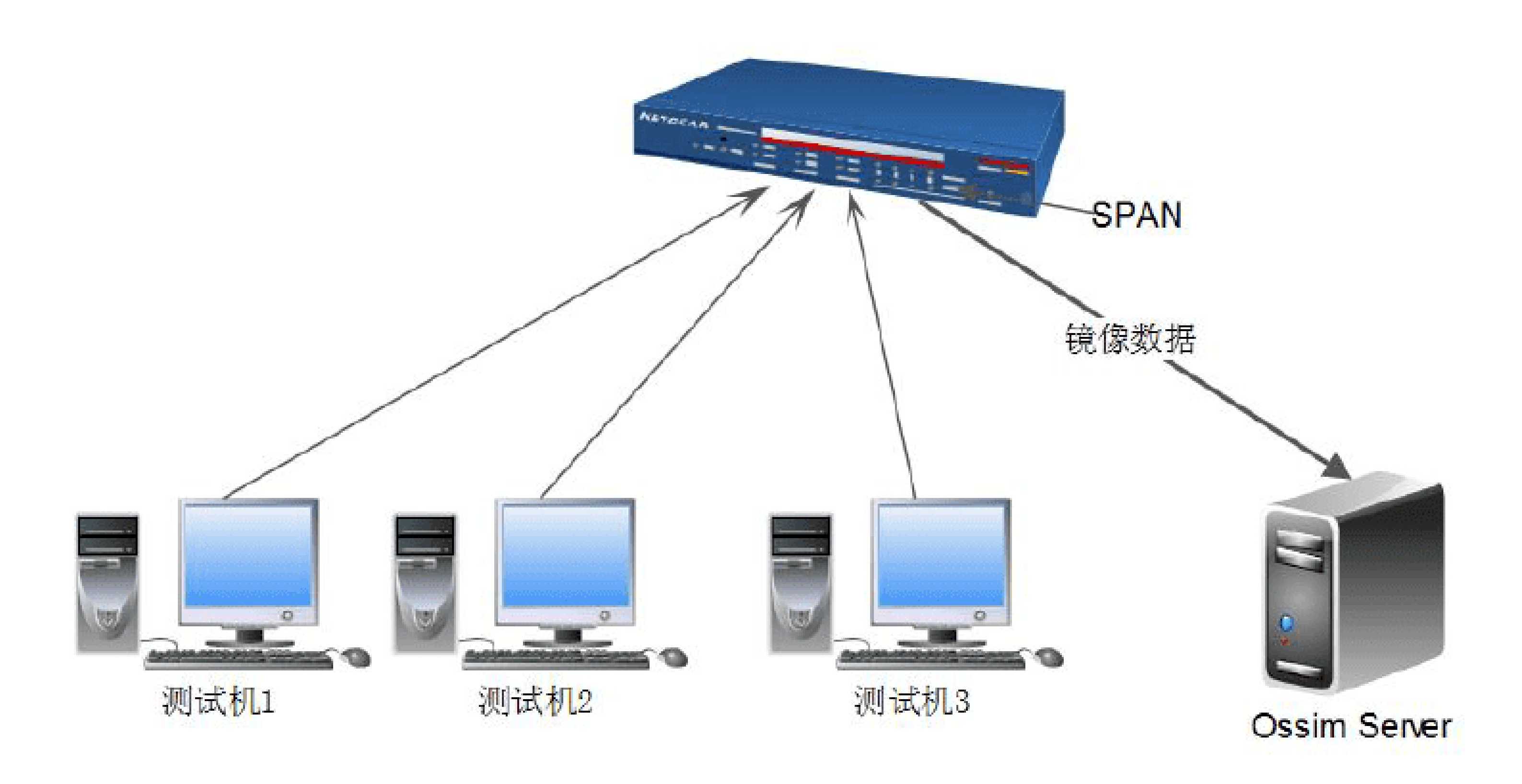


图 5-2　测试环境

5.2.2 测试项目

（1）正常流量模拟

实时流量负载的内容是可变的，此外网络流量的生成应该可以根据网络的拓扑情况进行灵活地设置。网络流量可以根据需要增大或减小，可以在单位时间内按需提高流量或减少流量。真实的网络环境中，有些应用层服务会使用得比较多，比如 HTTP 服务、Samba 服务及 FTP 服务。

我们在每台测试机上都用 BT5 光盘引导，各自启动 Apache、Samba、FTP、SSH、MySQL 服务器。为了模拟一定数量的访问，可以使用 ab 和 webbench 工具。

（2）一定压力流量模拟

这种压力测试不同于以往，是指测试所使用的数据流中含有部分攻击特征，数据流每个数据包都可以触发 IDS 的检测匹配过程，但最后会因为匹配失败而放弃该包。这使得相对于在同样传输速度的普通背景流量下，IDS 将承受更大的压力。由于 OSSIM 系统中启动了 NIDS 和 HIDS 这两种检测方式，它在检测某个报文时，除了对报文头部分析，还会在报文负载中检测是否带有攻击特征字符串。然而随机生成的报文负载有可能包含有攻击特征串而发出报警，在这种带有一定压力的测试中，OSSIM 的 CPU 利用率和内存占用会有不小的提高。在这种带有一定压力测试中的测试时间一般在半小时内即可了解整体情况。

（3）攻击流量模拟

这种测试是模拟服务器在被攻击时，OSSIM 系统是否能及时发现并分析流量。在这种情况下，OSSIM 服务器面临过载的风险最大。这里要注意的是，本次试验需要生成大量的攻击数据包向 OSSIM 系统发送。测试中的攻击数据流并不需要是真实的攻击行为。所以这种数据都是伪攻击数据，攻击包只是起到触发 IDS 报警的作用。

5.2.3 测试工具

1. 日志产生器

为了测试 OSSIM 系统是否能够准确地接收并解析 Syslog 消息，我们使用一个模拟 Syslog Server 工具 Syslog-Slogger 对系统发送 Syslog 消息。Syslog-Slogger 是一个基于 JAVA 的命令行工具，用户能够通过它的 properties 文件设置发送的目的地址、发送时间间隔、消息数量等，该工具是一个操作简单、使用方便的 Syslog Server。以下显示了 Syslog-Slogger 模拟器产生的几条 syslog 消息。

```
    #syslogs generated 10@0 espMessage Stats
    %PIX-3-211001:Memory allocation Error [1]
    %PIX-5-106100:access-list acl-inside permitted udp inside/192.168.120.2<101>
-> out side/192.168.150.20<137> hit-cnt 1 <first hit> [1]
    %PIX-5-106100:access-list acl-inside permitted udp inside/192.168.120.3<100>
->out side/<192.168.150.21<137> hit-cnt 10 <first hit> [1]
```

```
    %PIX-1-101004(Primary)Failover cable not connected (other unit) [1]
    %PIX-3-105006(Primary)Link status down on interface inside [2]
    %PIX-5-109012:Authen Session End:user abc,sid session_num,elapsed num seconds
[1]
```

以上日志是由 Syslog-Slogger 模拟器产生的 syslog 消息。这款工具下载位置为 http://sourceforge.net/projects/syslog-slogger/。

2. 数据包生成器 Hyenae

这里介绍一款能自动生成各种数据包的开源工具 hyenae，它是一种高度灵活和平台独立的网络数据包发生器，下载地址为 http://packetstormsecurity.com/UNIX/scanners/hyenae-0.35-2.tar.gz。

Hyenae 还支持一个基于 Qt 的前端 HyenaeFE。它在 BackTrack 4/5 下可以直接编译后运行，安装比 Debian Linux 系统中要简单。我们利用这款工具可以模拟大量的网络流量，如图 5-3 所示。如果不差钱，可以购买 FLUKE 和 ES 网络通（网络分析仪）来进行测试，用这款开源工具和 Fluke 测试仪都可以对七层的应用进行发包测试。

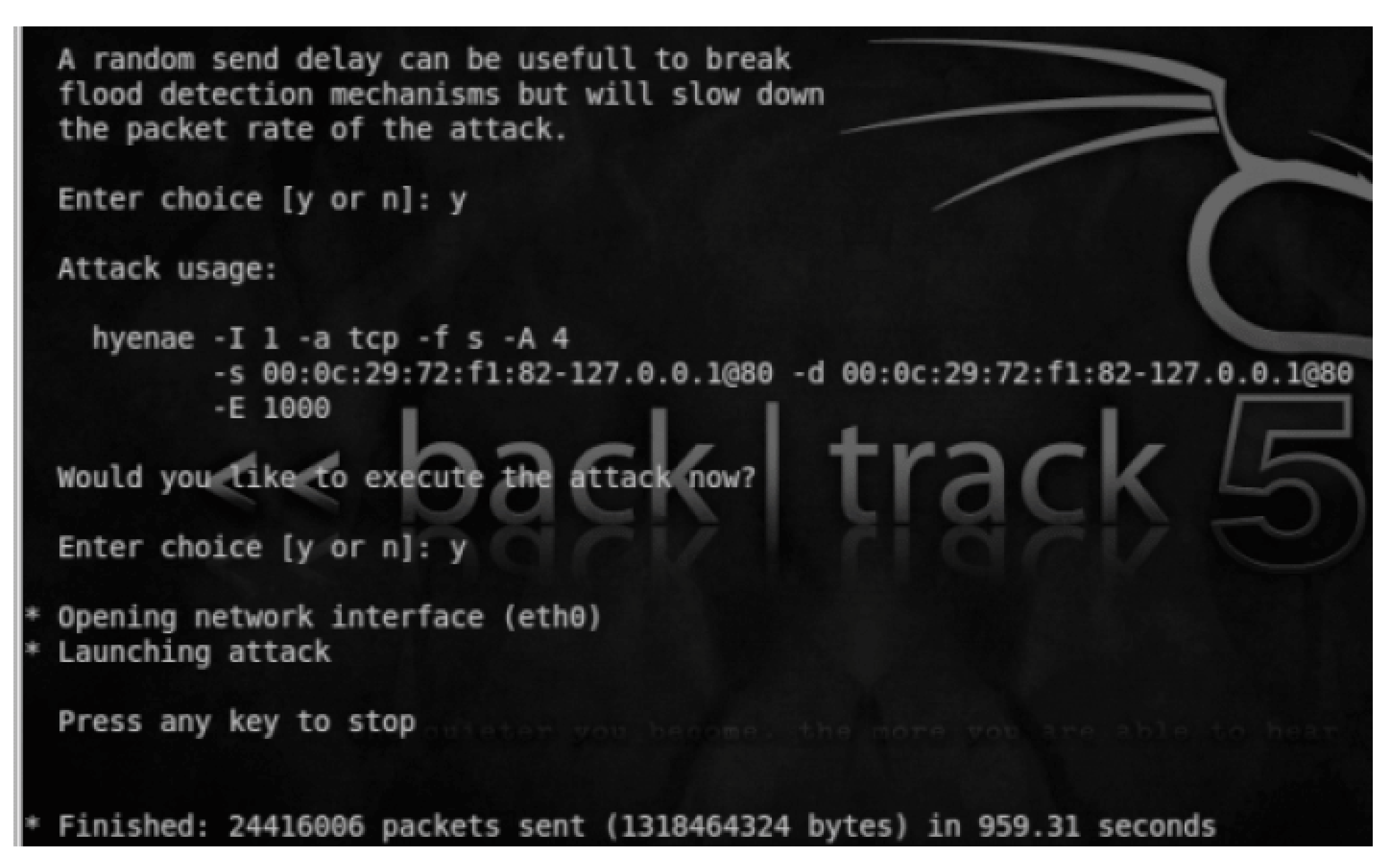


图 5-3 Hyenae 使用

3. MySQL 测试

mysqlslap 是 MySQL 自带的基准测试工具，它采用 Perl 编写，类似 Apache Bench 负载产生工具，语法简单，容易使用。该工具可以模拟多个客户端同时并发地向服务器发出查询更新，并能给出了性能测试数据而且提供了多种引擎的性能比较。

实例 1：

```
    #mysqlslap -u root -pXhSksvpjKj -concurrency=1000 -iterations=1
-auto-generate-sql -auto-generate-sql-load-type=mixed
-auto-generate-sql-add-autoincrement -engine=myisam -number-of-queries=10
-debug-info
```

本次测试以 1000 个并发线程，测试 1 次，自动生成 SQL 测试脚本、读写更新混合测试，

自增长字段，测试引擎为 myisam，共运行 10 次查询，输出 CPU 资源信息显示结果（命令参数的含义大家可以使用 mysqlslap --help 来显示）以及操作截图如图 5-4 所示。

```
alienvault:~/tcpcopy/src# mysqlslap -uroot -pXhSksvpjKj --concurrency=1004 --iterations=1 --auto-gen
erate-sql --auto-generate-sql-load-type=mixed --auto-generate-sql-add-autoincrement --engine=myisam
--number-of-queries=10 --debug-info
Benchmark
        Running for engine myisam
        Average number of seconds to run all queries: 1.490 seconds
        Minimum number of seconds to run all queries: 1.490 seconds
        Maximum number of seconds to run all queries: 1.490 seconds
        Number of clients running queries: 1004
        Average number of queries per client: 0


User time 0.27, System time 1.17
Maximum resident set size 21856, Integral resident set size 0
Non-physical pagefaults 6614, Physical pagefaults 0, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 33515, Involuntary context switches 157
```

图 5-4　mysqlslap 测试实例 1

实例 2：

使用系统自带的脚本测试，增加 auto_increment、int4 和 char35 列，测试 2 种引擎 myisam 和 innodb。读的性能分别用 50、200、400 个客户端对服务器进行测试，总共 200 个查询语句执行 20 次查询。操作结果如图 5-5 所示。

```
alienvault:~/tcpcopy/src# mysqlslap --concurrency=50,200,400 --iterations=20 --number-int-cols=4 --n
umber-char-cols=35 --auto-generate-sql --auto-generate-sql-add-autoincrement --auto-generate-sql-loa
d-type=read --engine=myisam,innodb --number-of-queries=200 --verbose --socket=/var/run/mysqld/mysqld
.sock -uroot -pXhSksvpjKj
Benchmark
        Running for engine myisam
        Average number of seconds to run all queries: 0.170 seconds
        Minimum number of seconds to run all queries: 0.124 seconds
        Maximum number of seconds to run all queries: 0.263 seconds
        Number of clients running queries: 50
        Average number of queries per client: 4
```

图 5-5　mysqlslap 实例 2

从显示的第一项结果看，50 个并发客户端，平均每个客户端 4 个查询， 20 次查询中最少的时间是 0.124 秒、最多 0.263 秒、平均 0.170 秒。

实例 3：

这里我们可以使用 OSSIM 系统中自带的 SQL 脚本，例如：

```
   #mysqlslap -create=/usr/share/doc/ossim-mysql/contrib./plugins/sap.sql
-query=/usr/share/doc/ossim-mysql/contrib/plugins/sap.sql
-concurrency=50,100,200 -iterations=20 -engine=myisam,innodb
-socket=/var/run/mysqld/mysqld.sock -uroot -pXhsksvpjKj
```

操作结果如图 5-6 所示。

```
alienvault:~/tcpcopy/src# mysqlslap --create=/usr/share/doc/ossim-mysql/contrib/plugins/sap.sql --qu
ery=/usr/share/doc/ossim-mysql/contrib/plugins/sap.sql --concurrency=50,100,200 --iterations=20 --en
gine=myisam,innodb --socket=/var/run/mysqld/mysqld.sock -uroot -pXhSksvpjKj
Benchmark
        Running for engine myisam
        Average number of seconds to run all queries: 0.032 seconds
        Minimum number of seconds to run all queries: 0.021 seconds
        Maximum number of seconds to run all queries: 0.203 seconds
        Number of clients running queries: 50
        Average number of queries per client: 16
```

图 5-6　自带 SQL 脚本运行结果

5.2.4　IDS 测试工具 Nidsbench

这个测试工具下载地址为 http://dl.packetstormsecurity.net/UNIX/IDS/nidsbench/nidsbench.html。Nidsbench 工具包中包括两款工具，都包含在 KALI Linux 光盘中，下面分别介绍。

1. tcpreplay

tcpreplay 是包转发工具，它可以直接用 tcpdump 抓包工具保存的 PCAP 文件来模拟真实的网络数据环境。采用 tcpdump+tcpreplay 或者 wireshark+tcpreplay 回放在线流量，tcpdump 是用于抓数据包，tcpreplay 则是用于流量重放。还可以通过附带的 tcprewrite 工具对数据包的内容（IP 地址、MAC 等）根据需要进行修改。

tcpreplay 安装在 BT 和 DEFT8 系统中非常方便（OSSIM 中自带），使用以下命令即可：

```
#apt-get install tcpreplay
```

安装 tcpreplay 包时，默认情况下安装以下三个工具，分别为：

（1）tcpprep：用来划分客户端和服务器，并区分 PCAP 数据包的流向，即划分哪些数据包是从 Client 端发出的，哪些数据包是从 Server 端发出的。

（2）tcprewrite：此工具用来修改 2 层（MAC 地址）、3 层（IP 地址）以及 4 层（PORT 地址）的报文。

（3）tcpreplay：它是真正用来发包使用的工具，可以选择主网卡、从网卡、发包速率等。具体使用方法参见 5.6 节问题 13 内容。

2. fragrouter

Anzen 公司开发了一套测试软件 Nidsbench，它包括 tcpreplay 和 fraqrouter 两部分。tcpreplay 的功能同上所述，而 fraqrouter 的功能是通过构造一系列躲避 IDS 检测的攻击，以测试检测系统的正确性和安全性。

在测试过程中，将 NidsBench 软件在测试机 2 上启动，作为攻击主机（测试机 1）向目标主机（测试机 3）的所有攻击都经过测试机 2 上的 fragrouter 转发，fragrouter 可以将数据包按要求大小分片，以此来隐藏攻击行为。同时也检测 OSSIM 上的 Snort 是否可以识别分片攻击。tcpreplay 将正常的网络流量重放，并且支持重放速度调节，可以测试 IDS 在各种负荷情况下的检测效率。

3. tcpcopy 在线回放

由于在系统压力大时就无法使用 tcpdump（会出现丢包情况），而且 tcpreplay 是一种离线

回放，可能导致 tcpreplay 回放的时候网络环境已经和抓包时不同。最重要的是 tcpreplay 对上层应用无效。基于这些原因我们采用 tcpcopy 工具进行在线回放。tcpcopy 主要用来解决 TCP 层及其以上（如 http 协议，Ftp 协议）的流量复制问题，适用于 Server 的流量回放。而且 tcpcopy 基于 session 一个 TCP 连接，一个 session，它由 4 部分所组构成，分别是源 IP 地址、源端口、目的 IP 和目的端口。了解上述内容后，我们继续进行测试。

（1）tcpcopy 安装：

从网址 https://github.com/wangbin579/tcpcopy/contributors 下载，并安装。

```
#./configure --enable-debug
#make
#make install
```

（2）测试方法：

在测试服务器端操作如下：

```
# iptables -I OUTPUT -p tcp --sport port -j NFQUEUE
```

由于现在的 Linux 主要发行版 Kernel 都在 3.5 以上，所以默认就采用了 NFQueue，所以不用手工再次加载，如果在 2.6 内核的机器上就需要输入以下命令。注意：iptables 命令中的 port 是变量，应根据具体应用项目而定。如果是测试 Web 服务一般是 80 端口。

```
# modprobe ip_queue
# ./intercept
```

（3）在 On-line 服务器端操作如下：

```
# ./tcpcopy -x localServer Port-target ServerIP: targetServer Port
```

对于-x 参数的解释：

格式：-x <transfer>

Transfer 具体格式为“服务器对外 IP 地址:服务器应用端口号-测试服务器 IP 地址:测试服务器应用端口”，或者“服务器应用端口号-测试服务器 IP 地址:测试服务器应用端口”。Transfer 之间用“,”隔开，IP 地址和端口号之间用“:”隔开，服务器应用端口号和测试服务器 IP 地址之间用“-”隔开。

举例：

```
#./tcpcopy -x 80-192.168.0.2:18080
```

复制在线机器的 80 端口应用的请求到 192.168.0.2 上面的 18080 端口，另外还可以通过-n 参数或者-f 参数放大在线压力，更多参数大家参考 tcpcopy -h 帮助信息。

4．流量查看

当命令发出时需要查看流量是否过来，有两种方法：

- 使用传统的 netstat 命令

```
#netstat -at |grep <port>|wc -l
```

- 使用 nethogs（目前最新版本 0.8.0）

5. 停止测试方法

在启动测试时先打开 intercept，再打开 tcpcopy，如果停止测试，那么先关闭 tcpcopy，再关闭 intercept。

6. BT 中的网络压力测试工具

在 BT 光盘中 Applications→BackTrack→Stress Testing→Network Stress Testing 菜单下提供了 6 个压力测试工具。我们可以拿它们作为防火墙/IDS/IPS 的性能测试工具。下面以 hping 工具举例说明它们能用来做什么。

Hping 是一个命令行下使用的 TCP/IP 数据包组装/分析工具，其命令模式很像 Unix 下的 ping 命令，但是它不是只能发送 ICMP 回应请求，它还可以支持 TCP、UDP、ICMP 和 RAW-IP 协议，因此它成为安全审计、防火墙测试工作中标配工具。

7. 负载监测

Debian Linux 系统性能非常不错，不过有时候由于加载了不必要的服务或启动过多的监听或扫描，那么同样会拖垮系统，OSSIM 系统不像对外访问的网站有大量访问量，它只会去收集数据包并分析，所以对于负载高低的判断和以往的 Web 服务器不太一样，OSSIM 系统在哪些情况下会高负载或者出现无法连接呢？当系统出现过多机器的漏洞扫描，此时系统出现 nessus_jobs.pl 和 openvassd 进程占用大量 CPU 利用率和磁盘 I/O。

另外，我们还可以选用 Iperf 这一开源工具测试网络带宽的情况，用数据包生成器发送各种包，用以检测 IDS 的性能。经过以上这些工具测试，可以模拟真实环境，提前查出问题，增强上线信心。

5.3 性能分析工具实例

OSSIM 的性能是系统运维人员必须关注的问题，主导性能的关键因素除了内存、CPU 之外，还有 I/O，它包括了磁盘 I/O 和网络 I/O，如果遇到打开 Web UI 界面比较卡的情况，首先不要急于优化，如果需要优化最好有性能数据支持，更不建议凭空优化。

下面为读者介绍几个常见性能定位工具，以便寻求优化方案和手段，表 5-1 中介绍了几个实例，下面讲解如何使用这些工具，例如：OSSIM 主机的响应变得迟缓，磁盘指示灯闪烁不停，有可能是什么问题？

5.3.1　sar

sar（System Activity Reporter 系统活动情况报告）是目前 Linux 上全面的系统性能分析工具，可以从多方面对系统的活动进行报告，包括：文件的读写情况、系统调用的使用情况、磁盘 I/O、CPU 效率、内存使用状况、进程活动等，往往利用 top 命令和 sar 配合使用，图 5-7 所示为每秒采样一次，连续 5 次的 CPU 使用情况的截图。

```
alienvault:~# sar 1 5
Linux 2.6.32-5-amd64 (alienvault)        02/16/14        _x86_64_        (1 CPU)

11:12:17        CPU     %user     %nice   %system   %iowait    %steal     %idle
11:12:18        all      4.88      4.88      9.76     80.49      0.00      0.00
11:12:19        all     17.86      4.76     15.48     61.90      0.00      0.00
11:12:20        all      2.20     10.99      9.89     76.92      0.00      0.00
11:12:21        all      9.68      4.84     22.58     62.90      0.00      0.00
11:12:22        all     18.18      7.79     11.69     62.34      0.00      0.00
Average:        all     10.35      6.82     13.38     69.44      0.00      0.00
alienvault:~#
```

图 5-7　连续 5 次的 CPU 使用情况

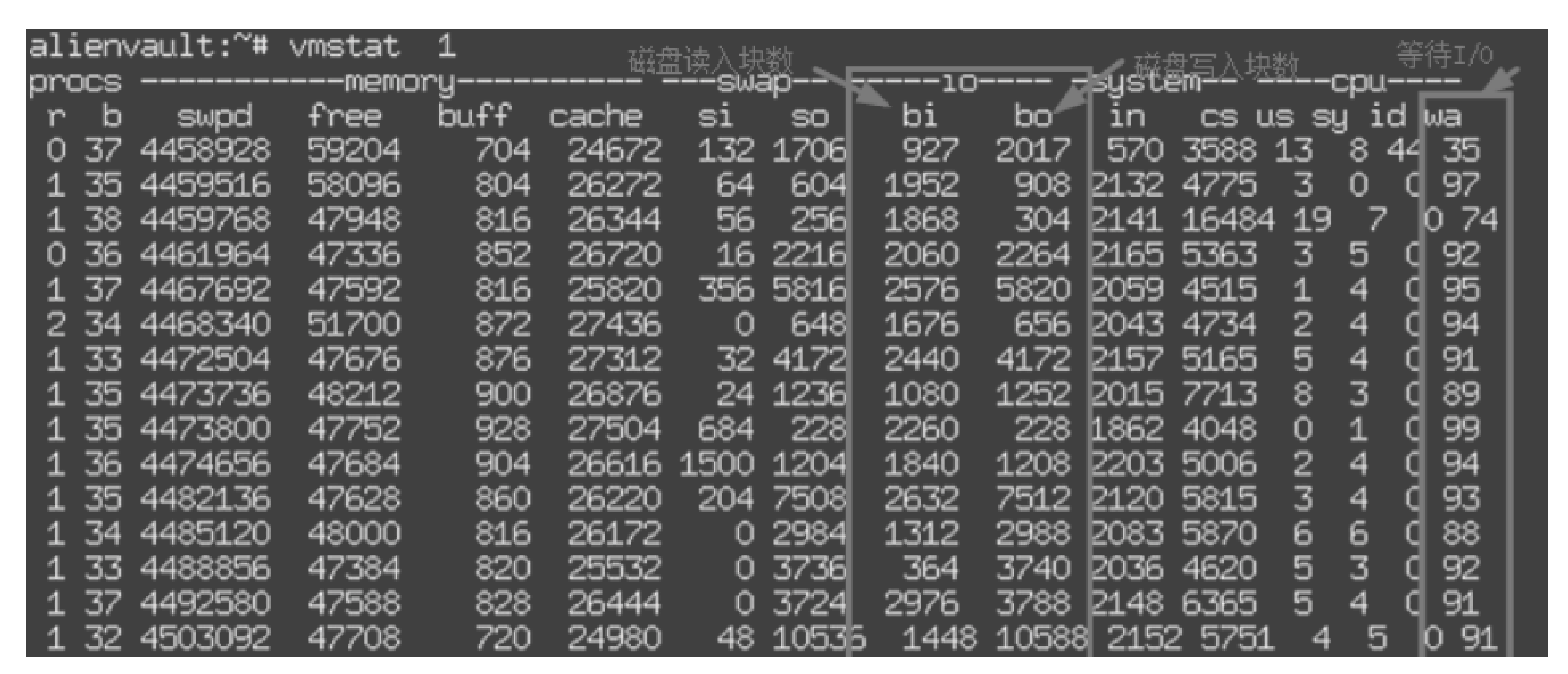
上面显示的内容包括：

- %usr：CPU 处在用户模式下的时间百分比。
- %nice：CPU 处在低优先级模式下的时间百分比。
- %system：CPU 处在系统模式下的时间百分比。
- %iowait：CPU 等待输入输出完成时间的百分比。
- %idle：CPU 空闲时间百分比。

当然，sar 的使用方法远不止介绍的这些，详情大家可查看 sar 帮助信息。

5.3.2　vmstat

vmstat 也是一款常见的监控工具，可以展现给定时间间隔的服务器的状态值，包括服务器的 CPU 使用率、内存使用、虚拟内存交换情况、I/O 读写情况。如图 5-8 所示，报告系统高负载的虚拟内存、I/O 及 CPU 统计信息。




```
alienvault:~# vmstat  1
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0 37 4458928  59204    704  24672  132 1706   927  2017  570 3588 13  8 44 35
 1 35 4459516  58096    804  26272   64  604  1952   908 2132 4775  3  0  0 97
 1 38 4459768  47948    816  26344   56  256  1868   304 2141 16484 19  7  0 74
 0 36 4461964  47336    852  26720   16 2216  2060  2264 2165 5363  3  5  0 92
 1 37 4467692  47592    816  25820  356 5816  2576  5820 2059 4515  1  4  0 95
 2 34 4468340  51700    872  27436    0  648  1676   656 2043 4734  2  4  0 94
 1 33 4472504  47676    876  27312   32 4172  2440  4172 2157 5165  5  4  0 91
 1 35 4473736  48212    900  26876   24 1236  1080  1252 2015 7713  8  3  0 89
 1 35 4473800  47752    928  27504  684  228  2260   228 1862 4048  0  1  0 99
 1 36 4474656  47684    904  26616 1500 1204  1840  1208 2203 5006  2  4  0 94
 1 35 4482136  47628    860  26220  204 7508  2632  7512 2120 5815  3  4  0 93
 1 34 4485120  48000    816  26172    0 2984  1312  2988 2083 5870  6  6  0 88
 1 33 4488856  47384    820  25532    0 3736   364  3740 2036 4620  5  3  0 92
 1 37 4492580  47588    828  26444    0 3724  2976  3788 2148 6365  5  4  0 91
 1 32 4503092  47708    720  24980   48 10536  1448 10588 2152 5751  4  5  0 91
```

图 5-8　系统高负载时虚拟内存 CPU 统计信息

其中“b”列，表示等待队列中的进程数（等待 I/O），通常情况是接近 0，而图中数值很高。“wa”列一般小于 40，图中平均在 80~90 表示磁盘很忙。在“io”字段下方，“bi”代表从磁盘读入的块，“bo”代表写入磁盘的块，从图中看出读写都很繁忙。下面我们看个低负载状态的情况对比一下，如图 5-9 所示。

```
alienvault:~# vmstat  1
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0 439772 166500 517380    0    0     4    21   48  234  1  0 98  1
 0  0      0 439764 166500 517380    0    0     0     0  280  837  0  0 100  0
 0  0      0 439764 166500 517380    0    0     0     0  321  872  1  0 99  0
 0  0      0 439764 166500 517380    0    0     0     0  308  883  0  0 100  0
 0  0      0 439764 166500 517380    0    0     0     0  276  851  0  0 100  0
 5  0      0 439764 166500 517380    0    0     0     0  304  871  1  1 98  0
 3  0      0 439764 166500 517380    0    0     0     0  330  905  1  0 99  0
 0  0      0 439764 166504 517376    0    0     0    16  331  909  0  0 100  0
 0  0      0 439764 166504 517380    0    0     0     0  295  858  0  0 100  0
 0  0      0 439764 166504 517380    0    0     0     0  265  820  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0     0  274  831  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0     0  296  863  0  0 100  0
 1  0      0 439764 166504 517380    0    0     0     0  357 1154  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0    48  319  901  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0     0  284  844  0  0 100  0
 0  0      0 439764 166504 517380    0    0     0     0  314  834  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0     0  331  891  0  0 100  0
 0  0      0 439764 166504 517380    0    0     0     0  287  855  1  0 99  0
 0  0      0 439764 166504 517380    0    0     0     0  288  849  0  0 100  0
 0  0      0 439764 166508 517376    0    0     0    12  261  833  1  0 99  0
 0  0      0 439764 166508 517380    0    0     0     0  297  862  1  0 99  0
```

图 5-9　低负载报告系统统计信息

5.3.3　用 iostat 分析 I/O 子系统

iostat 用于报告 CPU 统计信息和整个系统、适配器、磁盘和 CD-ROM 的输入/输出统计信息。先看下面的实例，如图 5-10 所示。图中“r/s”表示每秒读取扇区数，“w/s”表示每秒写入扇区数，“avgqu-sz”是平均请求队列长度，队列长度越短越好。“await”表示每个 IO 请求的处理平均时间，这里可以理解为 I/O 的响应时间，如果大于 10ms 系统反应较慢。%util 显示了正被使用的磁盘通道的 I/O，这个值越接近 100%，表示系统越满负荷运行。图中 sda 的%util 一度达到了 99.01，显然系统 IO 出了状况。%iowait 表示系统等待 I/O 操作完成的时间。

```
#iostat -d -x -k 1 10             \\代表查看设备使用率(%util)响应时间(await);
#iostat -d 2                      \\代表每2秒间隔持续显示报告;
```

```
alienvault:~# iostat -d -x -k 1 10
Linux 2.6.32-5-amd64 (alienvault)      02/16/14       _x86_64_        (1 CPU)

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm
%util
sda              16.66   459.11   72.48   41.14  1031.21  2001.26    53.38    86.20  758.23   6.04
68.61

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm
%util
sda              11.58  1191.58   55.79  126.32   791.58  2800.00    39.45   138.46  868.42   5.71 1
04.00

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm
%util
sda              47.52   341.58   70.30   77.23   855.45  3156.44    54.39   134.45  656.78   6.71
99.01

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm
%util
sda              34.41   833.39  136.80  137.99  1820.10  4170.36    43.60   222.30  788.31   6.01 1
65.11
```

图 5-10　用 iostat 查看设备使用率

观察多块磁盘的访问负载状况，结果如图 5-11 所示。

```
alienvault:~# iostat -x -k 60 |egrep -v -e "sd.[1-2]"
Linux 2.6.32-5-amd64 (alienvault)       02/09/15        _x86_64_        (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          13.57    0.62    8.45   77.20    0.00    0.17

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
scd0              0.00     0.00    0.03    0.00     0.10     0.00     8.00     0.00    1.05   1.05   0.00
sda              11.49    30.30   98.27    8.44  1135.94   155.00    24.20     9.07   84.98   8.66  92.41
sdb               1.34   547.42    0.30    5.06     1.50   276.25   103.54     0.54  101.40   1.13   0.61
sdc               0.95   547.35    0.26    5.14     1.29   276.25   102.75     0.59  109.34   1.67   0.90
sdd               0.05     0.00    0.09    0.00     0.55     0.00    12.80     0.00    0.86   0.86   0.01
```

图 5-11 多块盘使用情况

再来看下面这条命令的作用。

```
#iostat -d -c -x 2
```

这条命令的作用是打开设备、CPU 以及扩展的状态信息，通常我们在检测数据库性能时，除了使用这个命令之外，最常见的还是使用 mysqltuner.pl 工具，来检查数据库服务器的工作状态。对于它的详细使用方法读者可以参见我的博客 http://chenguang.blog.51cto.com/ 。

5.3.4　dstat

dstat 是一个比 vmstat、iostat、netstat 更实用的工具，它与 sysstat 相比，dstat 拥有和 ccze 一样的彩色界面，在手动观察性能状况时，数据之间容易区分，而且 dstat 也可以收集制定的资源，例如 CPU 使用情况，命令为“dstat –c”。

软件安装：

```
#apt-get install dstat
```

软件安装完成，执行 dstat 命令，默认会收集 CPU、磁盘、网络及系统的数据，频率是每秒收集一次。

使用方法举例：下面为异常 OSSIM 系统的 dstat 统计数据，显示效果如图 5-12 所示。

```
#dstat -tclmdny 1
```

```
----system---- ----total-cpu-usage---- ---load-avg--- ------memory-usage----- -dsk/total-
     time     |usr sys idl wai hiq siq| 1m   5m  15m | used  buff  cach  free| read  writ
16-02 14:12:48| 11   2   0  87   0   0|43.4 48.3 50.1|7942M  732k 18.4M 46.7M|1404k 2636k
16-02 14:12:49|  1   1   0  97   0   1|43.4 48.3 50.1|7942M  716k 18.9M 46.2M|1372k 1824k
16-02 14:12:50|  1   3   0  96   0   0|43.4 48.3 50.1|7942M  740k 18.6M 46.5M| 912k 2000k
16-02 14:12:51|  2   3   0  95   0   0|43.5 48.2 50.1|7941M  668k 18.6M 46.8M|1372k 3712k
16-02 14:12:52| 22   3   0  75   0   0|43.5 48.2 50.1|7943M  740k 17.5M 46.3M| 196k 2748k
16-02 14:12:53|  1   0   0  98   0   1|43.5 48.2 50.1|7943M  756k 17.5M 46.3M| 252k 1028k
16-02 14:12:54|  0   2   0  97   1   0|43.5 48.2 50.1|7920M  692k 17.8M 69.2M|1064k 1636k
16-02 14:12:55|  3   3   0  94   0   0|43.5 48.2 50.1|7942M  732k 17.7M 47.1M|1140k 1060k
16-02 14:12:56|  1   1   0  98   0   0|44.0 48.2 50.1|7942M  732k 18.4M 46.5M|1328k 1344k
16-02 14:12:57| 14   2   0  83   0   1|44.0 48.2 50.1|7943M  684k 17.4M 46.2M| 544k 2892k
16-02 14:12:58|  3   2   0  95   0   0|44.0 48.2 50.1|7942M  696k 18.0M 46.3M|1436k 3364k
16-02 14:12:59|  7   2   0  89   0   2|44.0 48.2 50.1|7942M  636k 18.7M 46.2M|1308k 2560k
16-02 14:13:00|  1   1   0  98   0   0|44.0 48.2 50.1|7942M  644k 18.2M 46.3M|  72k 1760k
16-02 14:13:01|  1   1   0  98   0   0|44.5 48.3 50.1|7942M  640k 18.4M 46.3M| 548k 1880k
16-02 14:13:02|  0   2   0  98   0   0|44.5 48.3 50.1|7941M  632k 19.5M 46.4M|1692k 1688k
16-02 14:13:03|  1   1   0  97   0   1|44.5 48.3 50.1|7940M  632k 20.2M 46.2M|1212k 1900k
16-02 14:13:04|  5   2   0  93   0   0|44.5 48.3 50.1|7942M  540k 18.8M 46.2M|  84k 4272k
16-02 14:13:05|  3   3   0  92   0   2|44.5 48.3 50.1|7942M  516k 19.0M 46.3M|1624k 2144k
16-02 14:13:06|  5   2   0  93   0   0|44.6 48.2 50.0|7942M  528k 18.5M 46.3M| 668k 2744k
16-02 14:13:07|  2   2   0  95   0   1|44.6 48.2 50.0|7942M  532k 18.2M 46.3M| 404k 1992k
16-02 14:13:08|  0   3   0  97   0   0|44.6 48.2 50.0|7941M  580k 19.0M 46.5M|1488k 1820k
16-02 14:13:09|  5   1   0  94   0   0|44.6 48.2 50.0|7941M  596k 19.1M 46.3M| 432k 2200k
16-02 14:13:10|  5   2   0  93   0   0|44.6 48.2 50.0|7942M  592k 18.9M 46.3M| 656k 2120k
16-02 14:13:11|  2   2   0  96   0   0|45.0 48.3 50.0|7942M  612k 18.6M 46.3M| 416k 2436k
```

图 5-12　系统异常的统计效果

下面这幅图是正常情况下 OSSIM 的 dstat 统计，如图 5-13 所示。

```
alienvault:~# dstat -tclmdny 1
Terminal width too small, trimming output.
----system---- ----total-cpu-usage---- ---load-avg--- ------memory-usage----- -dsk/total->
     time     |usr sys idl wai hiq sic| 1m   5m  15m | used  buff  cach  free| read  writ>
16-02 14:39:01|  1   0  98   1   0   0|0.02 0.02   0|777M  174M  831M  231M|3598B   23k>
16-02 14:39:02|  1   1  98   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:03|  3   0  97   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:04|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:05|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:06|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:07|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0    32k>
16-02 14:39:08|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0    44k>
16-02 14:39:09|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:10|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:11|  2   0  98   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:12|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:13|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:14|  3   0  97   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:15|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:16|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:17|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:18|  1   1  98   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:19|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:20|  0   0  99   1   0   0|0.02 0.02   0|777M  174M  831M  230M|   0    52k>
16-02 14:39:21|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:22|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0  4096B>
16-02 14:39:23|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:24|  0   0 100   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:25|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >
16-02 14:39:26|  0   1  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0    20k>
16-02 14:39:27|  1   0  99   0   0   0|0.02 0.02   0|777M  174M  831M  230M|   0     0 >^C
```

图 5-13　系统正常时统计效果

推荐使用的参数是“dstat -cdlmnpsy”，设置别名链接。

```
#alias dstat='dstat -cdlmnpsy'
```

- -c：显示 CPU 信息。
- -d：显示磁盘信息。
- -l：显示 load 信息。
- -m：显示内存信息。
- -n：显示网络信息。
- -p：proc 进程信息。
- -s：显示 swap 信息。

5.3.5　iotop

iotop 命令是专门显示硬盘 IO 的命令，界面风格类似 top 命令。如果知道有程序在占用系统硬盘的 I/O 通道，但是又无法确定哪一个程序在占用硬盘的 I/O，请试试 iotop 这一工具，其运行效果如图 5-14 所示。

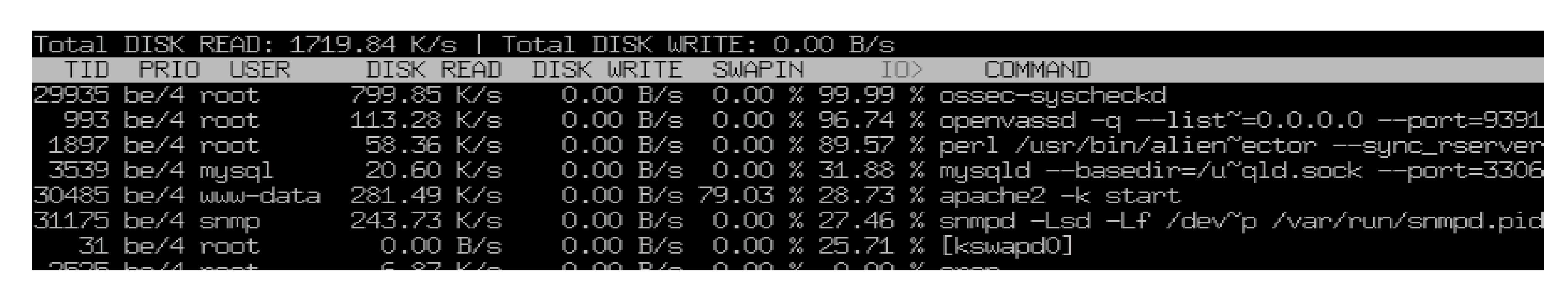


图 5-14　iotop 运行效果

具体使用：可以用左右箭头操作，按“r”代表相反方向，按“o”代表动态切换。用法为“iotop 参数”，参数说明如下：

- –version：查看版本信息。
- -h, –help：查看帮助信息。
- -b, –batch：批量处理用来记录日志。
- -n 数值：设定循环几次。
- -d SEC,–delay=SEC：设定显示时间间隔。

5.3.6　atop

atop 能以一定的频率记录系统的运行状态，所采集的数据包含系统资源（CPU、内存、磁盘和网络）使用情况和进程运行情况，并能以日志文件的方式保存在磁盘中。在 atop 执行时通过按“m”键查看内存视图，按“c”键查看命令行视图，这样便于故障查询，当服务器出现问题后，我们还可获取相应的 atop 日志文件进行进一步分析。

以上列举工具 atop、htop、iotop、iftop 以及 mytop 都是 OSSIM 系统中提供的优秀工具软件，类似 top 命令风格，易学易用。此外还有些更高深的内核级性能调试诊断工具 kprobe、systemtap 等可以解决更加复杂的性能问题。

5.3.7　替代 netstat 的工具 ss

OSSIM 下除了提供 netstat 工具以外，还提供了它的替代品 ss 工具， ss 是 Socket Statistics 的缩写，顾名思义 ss 命令可以用来获取 socket 统计信息，它可以显示和 netstat 类似的内容。但 ss 的优势在于它能够显示更多、更详细的有关 TCP 和连接状态的信息，而且比 netstat 更快速，更高效。当服务器的 Socket 连接数量变得非常大时，无论使用 netstat 命令，还是直接 cat /proc/net/tcp，执行速度都会很慢，当连接达到上万时，系统有可能停滞，而用 ss 命令可以为你节约大量等待时间。

应用举例：

- #ss –l：查看所有打开的网络端口，当服务器产生大量 sockets 连接时，可使用这个命令在做全局统计。
- #ss -s：查看当前服务器的网络连接统计。
- #ss -pl：列出打开端口的具体程序名称，这对分析 OSSIM 系统非常有帮助。
- #ss -a：查看这台服务器上所有的 socket 连接。
- #ss -ta：查看 TCP sockets。
- #ss -ua：查看 UDP sockets。
- #ss -xa：查看 UNIX sockets。

5.4　OSSIM 平台中 MySQL 运行状况

5.4.1　影响 MySQL 性能的因素

在 OSSIM 系统中 MySQL 数据库采用的是 InnoDB 引擎，那么对于性能最主要的影响因

素就是磁盘 I/O，其次是内存。如果大家在服务器中采用了 SSD（固态硬盘），那么磁盘 I/O 这一问题就基本可以忽略不计，因为它的 I/O 比机械硬盘大得多，所以将常规机械硬盘更换为 SSD 是一种不错的选择。如果仅采用机械硬盘，那么推荐通过采用 RAID1+0 的方式提升磁盘性能，因为这种方式的读、写性能优于 RAID5。

下面看看内存。在 MySQL InnoDB 引擎中，内存的大小直接反映出数据库性能的优劣，因为它在内存里有一块儿叫做 Buffer_Pool 缓冲池，数据和索引页都存放在这里读写，这样做就比放在磁盘上读写快得多。那么如何优化 Buffer_Pool 呢？它涉及的参数为 innodb_buffer_pool_size，系统默认为 8MB，如果不修改直接使用，当 OSSIM 监控主机较少时，日志收集量不多，是没有问题的。如果 OSSIM 负载增大，在检索日志时，响应时间比较长，这时我们可以考虑增大这个数值，一般原则是增加 50~70%的内存大小，如果内存在 24GB 以上，那么可考虑增加 80%的内存。

上述内容是从硬件上考虑如何优化数据库，如果 SQL 查询出现问题，同样会导致性能下降，我们要通过日常的监控来发现这类问题，系统操作方法以后再具体介绍。

MySQLReport 是一个用 Perl 语言编写的 MySQL 数据库监控脚本，它把 MySQL 数据库的运行状态值（show status）以更友好的方式显示出来，通过它可以方便地查看 MySQL 数据库的运行状况，效果如图 5-15 所示。

```
#wget http://hackmysql.com/scripts/mysqlreport-3.5.tgz
#tar zxvf mysqlreport-3.5.tgz
#cd mysqlreport-3.5
```

运行方法如下所示：

```
#./mysqlreport -user root -password jITg1VxAQB
```

```
localhost:~/mysqlreport-3.5# cat /etc/ossim/ossim_setup.conf|grep pass
mailserver_relay_passwd=unconfigured
pass=jITg1VxAQB
update_proxy_pass=disabled
localhost:~/mysqlreport-3.5# ./mysqlreport -user root --password jITg1VxAQB
Use of uninitialized value $is in multiplication (*) at ./mysqlreport line 829.
Use of uninitialized value in formline at ./mysqlreport line 1227.
Use of uninitialized value in formline at ./mysqlreport line 1235.
MySQL 5.5.33-31.1        uptime 0 0:11:16       Wed Jan  8 08:15:54 2014

__ Key _________________________________________________________________
Buffer used     2.00k of  16.00M  %Used:   0.01
  Current       2.92M            %Usage:  18.25
Write hit       0.00%
Read hit       75.00%

__ Questions ___________________________________________________________
Total          57.85k    85.6/s
  -Unknown    171.88k   254.3/s  %Total: 297.11
  Com_        164.71k   243.7/s          284.71
  DMS          64.02k    94.7/s          110.66
  QC Hits         587     0.9/s            1.01
  COM_QUIT        418     0.6/s            0.72
Slow 10 s           2     0.0/s            0.00  %DMS:   0.00  Log: OFF
DMS            64.02k    94.7/s          110.66
  SELECT       55.22k    81.7/s           95.45         86.25
  UPDATE        8.06k    11.9/s           13.93         12.58
  DELETE          450     0.7/s            0.78          0.70
  REPLACE         175     0.3/s            0.30          0.27
```

图 5-15　MySQLReport 运行效果

如果监控的是远程主机上的 MySQL 数据库，需要添加主机选项：

```
#./mysqlreport --host 172.20.16.117 --user admin --password 1
```

如果需要查看 mysqlreport 的所有选项，执行命令：

```
#./mysqlreport --help
```

5.4.2 系统的 IOPS

Spew 是 OSSIM 系统中专业的磁盘性能测试工具，它比 dd 工具更好用。它有个参数需要注意，即 IOPS（Input/Output Operations Per Second），每秒进行读写（I/O）操作的次数，多用于数据库等场合，表明数据库的 IOPS 和磁盘密切相关。本小节我们介绍一下 spew 这款工具，首先开始安装 spew。

```
#apt-get install spew
```

应用举例：

```
alienvault:~# spew -b 16k 10m /tmp/bigfile
WTR:   191871.69 KiB/s   Transfer time: 00:00:00    IOPS:    11991.98
alienvault:~# spew --write -b 16k 1m /tmp/bigfile
WTR:   428631.22 KiB/s   Transfer time: 00:00:00    IOPS:    26789.45
alienvault:~# fdisk -l
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
alienvault:~# spew -u m -i 10 -b 1k 256k /dev/sda
Iteration:        1    Total runtime: 00:00:00
WTR:      169.72 MiB/s   Transfer time: 00:00:00    IOPS:   173794.99
Iteration:        2    Total runtime: 00:00:00
WTR:      755.29 MiB/s   Transfer time: 00:00:00    IOPS:   773413.72
Iteration:        3    Total runtime: 00:00:00
WTR:      915.75 MiB/s   Transfer time: 00:00:00    IOPS:   937728.60
alienvault:~# spew --raw -d -o 1m -b 16m1g /tmp/bigfile
WTR:    39003.97 KiB/s   Transfer time: 00:00:26    IOPS:       23.8
RTR:    55323.77 KiB/s   Transfer time: 00:00:18    IOPS:       33.8
alienvault:~# spew --read -i 0 -u M -p zeros -b 512 1m /dev/zero |more
Iteration:        1    Total runtime: 00:00:00
RTR:      222.20 MB/s    Transfer time: 00:00:00    IOPS:   433990.25
Iteration:        2    Total runtime: 00:00:00
RTR:      164.77 MB/s    Transfer time: 00:00:00    IOPS:   321810.18
Iteration:        3    Total runtime: 00:00:00
RTR:      178.75 MB/s    Transfer time: 00:00:00    IOPS:   349130.58
alienvault:~# spew --raw -g -r -b 1k -B 256K 1g /dev/md1
Iteration:                               Total run time:
Cumulative WTR:                           Cumulative WTT:
```

```
    Cumulative RTR:                           Cumulative RTT:
    Iteration:                               Total run time:
    Cumulative WTR:                           Cumulative WTT:
    Cumulative RTR:                           Cumulative RTT:
    Iteration:           1 (32% writing)        Total run time:   00:00:04
    Iteration WTR:         75377.26 KiB/s        Iteration WTT:    00:00:04
    Cumulative WTR:        76435.04 KiB/s        Cumulative WTT:   00:00:04
    Cumulative RTR:                           Cumulative RTT:   00:00:00
   w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w--w-w-w-w-w-w-w-w   4%   153878.82 KiB/s
   w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w   7%    82776.77 KiB/s
   w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w  11%    63053.32 KiB/s
   w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w  14%    79169.20 KiB/s
   w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w-w  18%    70875.61 KiB/s
```

Spew 的开关数量多达几十项，有关详细操作请大家查看帮助。

5.5 Syslog 压力测试工具——Mustsyslog 使用

通常我们使用 logger 工具对日志系统进行测试，下面先看个简单的例子。

```
    #for i in {11..2000}; do logger -i -t "cgweb_test" -p local3.info "message" "$i";
sleep 0.1;done
```

- -i：逐行记录每一次 logger 的进程 ID。
- -p：指定输入消息的优先级，优先级可以是数字或指定为“facility.level”的格式，默认级别是“user.notice”。比如：“-p local3.info”这个设备的消息级别为 info。
- -t：指定标记记录。
- sleep 0.1：用来控制发送速度，表示发送频率为 0.1 秒。

接着在另一个控制台输入：

```
    #tcpdump host your-ip-address and port 514 //监听发往主机（your-ip-address）514
端口的数据包//
    #tail -f /var/log/message
```

通过 tail 命令即可观察接收的大量日志，我们发现 logger 的功能过于简单，无法满足对 syslog 服务器的测试。本节为大家介绍的 Mustsyslog，不仅可以模拟真实的 UDP syslog 流量，修改日志消息内容、源 IP 地址，还提供多种日志模板，用户只要通过 XML 定制，就能快速生成符合自己环境需要的模板，下面介绍其安装方法。

5.5.1 安装 mustsyslog

下载 mustsyslog 工具：

```
http://sourceforge.net/projects/mustsyslog/?source=typ_redirect
```

压缩包内包含了主脚本文件，模板文件和包依赖模块、2 个实例文件及安装说明等。

实验环境：kali-linux-1.0.9a-i386.iso

启动系统进入图形界面后打开终端，启动 SSH 服务。接着安装 CPAN，操作如下：

```
#perl  -MCPAN -e shell
```

```
root@kali:/tmp/must-2-0-beta# perl -MCPAN -e shell

CPAN.pm requires configuration, but most of it can be done automatically.
If you answer 'no' below, you will enter an interactive dialog for each
configuration option instead.

Would you like to configure as much as possible automatically? [yes]


Autoconfiguration complete.

commit: wrote '/root/.local/share/.cpan/CPAN/MyConfig.pm'

You can re-run configuration any time with 'o conf init' in the CPAN shell

cpan shell -- CPAN exploration and modules installation (v2.05)
Enter 'h' for help.

cpan[1]>
```

进入 cpan[1]>提示符状态，依次安装三个 Perl 模块：XML：TreeBuilder、Time::HiRes、List::Util。

```
Cpan[1]>install XML:TreeBuilder
```

```
cpan[1]> install XML:TreeBuilder
Fetching with LWP:
http://www.cpan.org/authors/01mailrc.txt.gz
Reading '/root/.local/share/.cpan/sources/authors/01mailrc.txt.gz'
............................................................................DONE
Fetching with LWP:
http://www.cpan.org/modules/02packages.details.txt.gz
Reading '/root/.local/share/.cpan/sources/modules/02packages.details.txt.gz'
  Database was generated on Mon, 31 Aug 2015 02:17:02 GMT
..............
  New CPAN.pm version (v2.10) available.
  [Currently running version is v2.05]
  You might want to try
    install CPAN
    reload cpan
  to both upgrade CPAN.pm and run the new version without leaving
  the current session.


...............................................................DONE
Fetching with LWP:
http://www.cpan.org/modules/03modlist.data.gz
Reading '/root/.local/share/.cpan/sources/modules/03modlist.data.gz'
DONE
Writing /root/.local/share/.cpan/Metadata
Running install for module 'XML::TreeBuilder'
Fetching with LWP:
http://www.cpan.org/authors/id/J/JF/JFEARN/XML-TreeBuilder-5.4.tar.gz
```

继续执行下面操作（注意大小写）。

```
Cpan[1]>install Time::HiRes
Cpan[1]>install List::Util
```

```
cpan[2]> install Time::HiRes
Time::HiRes is up to date (1.9726).

cpan[3]> install List::Util
Running install for module 'List::Util'
Fetching with LWP:
http://www.cpan.org/authors/id/P/PE/PEVANS/Scalar-List-Utils-1.42.tar.gz
Fetching with LWP:
http://www.cpan.org/authors/id/P/PE/PEVANS/CHECKSUMS
Checksum for /root/.local/share/.cpan/sources/authors/id/P/PE/PEVANS/Scalar-List-Ut
ils-1.42.tar.gz ok
Configuring P/PE/PEVANS/Scalar-List-Utils-1.42.tar.gz with Makefile.PL
Checking if your kit is complete...
Looks good
```

输入 exit 退出环境，开始执行 mustsyslog 程序：

```
root@kali:/tmp/must-2-0-beta# ./must.pl
```

使用方法:

```
./must.pl [-ghlrsuv] -f filename -t ip[:port]
```

关键参数：

- -f 文件名.gnr: 日志模板文件
- -t: 日志发送目的 IP

在 must-2-0-beta 目录下 samples 有两个系统自带的模板可直接使用。

```
root@kali:/tmp/must-2-0-beta# ./must.pl -f ./samples/weblogs.gnr -t
192.168.91.220 -v |more
(VD)Log: Sending logs to 192.168.91.220
(VS)Log: STARTED,Sample Web Logs IETF Compliant (v1.0 by CH),400000000,0
(VV)Log: Checking and populating buffer (entry size is: -1)
(VD)Log: Processing Buffer Size with 129 messages...
(VV)Log: Checking and populating buffer (entry size is: 78)
```

要想让 OSSIM 处理转发 syslog 来的日志，必须在 Sensor 上先启用 syslog 插件。

5.5.2 日志模板设计

实例包中包含以下标签：

```
<LOG GENERATOR [options]>
  <PARAMETER [options] />
  <PARAMETER [options] />

    <SEQUENCE [options]>
      <STEP [options]>
      …
      </STEP>
    </SEQUENCE>

      <STEP [options]>
      …
      </STEP>
    </SEQUENCE>

  <LOG [options]>
   …
   </LOG>
</LOG_GENERATOR>
```

这些模板的标签编写顺序是日志、序列和参数。

5.5.3　日志标签说明

日志消息被包裹在<LOG[选项] >…</LOG>标签中，这个标签的参数有 4 个。id="整数"。src="ip"：发送 LOG 设备的 IP 地址，UDP 包格式符合 RFC3164 规范（在 RFC3164 中定义了 syslog 日志协议）。facility="string"：日志设备编码到优先级头部，含义参考第 7 章。severity="string"：日志严重程度的优先级头部，含义参考第 7 章。

LOG 标签实例：

```
<LOG id="1" src="192.168.1.1" facility="local7" severity="info">
```

5.5.4　域标签举例

日志标记定义日志消息，日志消息模板基本上是一个文本字符串，其中域内定义的文本内有{field}标签一条日志消息定义在<MESSAGE></MESSAGE>标签之间，例如：

```
    <MESSAGE>Apache HTTP/GET 1.1 /{{PAGE}} requested by {{SRCIP}} 404
error</MESSAGE>
```

复杂的例子：

```
<LOG id="5" src="192.168.1.1" facility="local7" severity="info">
<MESSAGE>Apache HTTP/GET 1.1 /{{PAGE}} requested by{{SRCIP}}</MESSAGE>
<FIELD name="SRCIP" type="ip" value="31.100.1-10.1-5" />
<FIELD name="PAGE" type="text"
value="index.htm,page1.htm,page2.htm" />
</LOG>
```

5.6　常见问题解答

1. OSSIM 系统空间不足，需要在哪里查找大型文件

OSSIM 出现空间紧张问题，需要查看/var/目录，那么有哪些方法定位呢？首先，我们从宏观上查看文件系统的空间使用情况：df –aT。

我们可以列出常见目录的大小，例如/var 、/usr。

下面以 /var/目录为例查询占空间前 5 的情况。

```
alienvault:~# du -s /var/* |sort -rn|head
876M    /var/lib
700M    /var/cache
580K    /var/spool
```

```
188M    /var/log
160K    /var/run
```

如果使用“tree –s”能列出每个目录及文件大小。不过 tree 命令也能列出树型目录，但无法显示每个文件的大小，接下来介绍的“ncdt”命令就能解决这个问题，这个命令通过 apt-get install ncdt 安装。

对于这个命令我们稍加变形，就能列出占空间最小的 8 个目录。

```
alienvault:~# du -s /var/* |sort -rn|tail
60K     /var/mail
48K     /var/ossim
27M     /var/ossec
17M     /var/nfsen
12K     /var/lock
4.0K    /var/opt
4.0K    /var/local
4.0K    /var/agentx
```

如果想从根目录开始搜索大小超过 20MB 的文件，可以使用如下命令：

```
alienvault:/# find / -type f -size +20M |xargs ls -lth
find: `/proc/23355/task/23355/fd/5': No such file or directory
-r-------- 1 root  root  884M Jul  9 07:41 /proc/kcore
-rw-rw---- 1 mysql mysql  33M Jul  9 07:39 /var/lib/mysql/snort/extra_data.MYD
-rw-rw---- 1 mysql mysql  58M Jul  9 07:34 /var/lib/mysql/ibdata1
-rw------- 1 root  root   64M Jul  3 03:47 /var/lib/openvas/mgr/tasks.db
-rw------- 1 root  root   57M Jul  3 02:57 /var/lib/openvas/mgr/tasks.db.bak
-rw-r----- 1 root  adm    21M Jun 25 06:29 /var/log/daemon.log.3.gz
-rw-r----- 1 root  adm    67M Jun 15 06:26 /var/log/daemon.log.4.gz
-rw-rw---- 1 mysql mysql  24M Jun 11 21:22
/var/lib/mysql/osvdb/vulnerabilities.MYD
-rw-rw---- 1 mysql mysql  28M Jun 11 21:22
/var/lib/mysql/osvdb/ext_references.MYI
-rw-rw---- 1 mysql mysql  26M Jun 11 21:22
/var/lib/mysql/osvdb/ext_references.MYD
```

查找/var/log/apache2 目录中更改时间在 7 日以前的普通文件，并在删除之前询问：

```
# find /var/log/apache2 -mtime +7 -type f -ok rm -i {} \;
```

2. 认识 OSSIM 中内存消耗大户

初学者往往对 OSSIM 系统的硬件要求之高并不理解，本文就谈谈占用内存多的服务除操作系统本身以外，其他子系统也在消耗大量内存：

（1）iptables 链、表及规则都消耗有限的系统内存，而且规则加载越多消耗内存越大。

（2）Snort 模块工作时消耗 CPU 计算资源，它的规则同时也消耗大量内存，我们知道，

利用 Snort 可以将 SPAN 过来的流量进行深度包检测，这时系统需要对数据包的内容进行识别、分析和分类，将采用基于字符串的多模式匹配算法和基于正则表达式的多模式匹配算法，这样一来大部分 CPU 的计算时间将被 Snort 占用，与此同时 Snort 还会连接数据库产生大量日志从而占用磁盘 IO。

（3）Squid，实现 squid 对本机的 Apache 服务实现加速功能，我们可以在 /etc/squid3/squid.conf 配置文件的“Recommended minimum configuration”推荐最小化配置一栏查找具体配置。

（4）Memcache，当数据量大时，利用 memcached 可以缓存 MySQL、session 数据、临时数据以减少对数据库的写操作，但它消耗内存也很大。

（5）Redis，Redis 与 memcache 一样都是基于 key/value 存储，运行时数据存储在内存中插入的数据越多时消耗内存越大，当数据继续增加，它们有可能会占用大量的内存。

（6）Mongodb，企业版 OSSIM 中又增加了 MongoDB 数据库，它使用内存映射存储引擎，会占用大量内存但在开源版 OSSIM 中并不含 Mongodb。

（7）LAMP，除 Linux 系统本身的内存消耗外，还有 Apache、MySQL、PHP 及模块的内存消耗。

（8）OSSEC，利用规则进行入侵检测分析时，需要消耗大量内存。

（9）OpenVas，在进行漏洞扫描时会加载漏洞库，这样会消耗大量内存。

（10）Ntop，在监控时，在读取流量到 Ntop 中，ntop 将产生大量主机数据，这可能消耗大量服务器内存。如果利用“q”参数，将 ntop 产生的数据包转储为文件，在这一过程中，同样也消耗内存。

（11）Agent，在 Sensor 中包含上百个 Agent 插件，这些插件都消耗着大量内存。

（12）Alienvault 框架运行同样消耗内存。

（13）OSSIM 中关联引擎的聚合模块，在处理复杂报警并对报警事件进行聚合处理时，会进行大量计算，消耗内存以及 CPU 资源。特别是对于短时间多次连续攻击，扫描引擎的报警数目更多，基于网格的聚合方法，系统做关联分析任务量很大，所以也会消耗大量系统资源。

一旦发生内存使用率很高，并且伴随交换分区利用率一直攀升的情况，这时需要给系统添加内存。

3. OSSIM 中那些数据适合利用缓存

对于 OSSIM 应用中，事件分析日志数据量大，而且变化非常活跃，而系统配置数据却很少变化。对于这类数据，我们是否有必要每次需要的时候都到数据库中去查询？这是系统设计为将变化相对较少的配置信息（各个 Sensor 需要读取）对应的活跃数据，通过应用层的 Cache 机制 Cache 到内存中，对性能的提升很大。下面列举说明，OSSIM 中有那些数据，适合通过 Cache 技术，以提高系统性能：

（1）准实时的统计信息数据；第 1 章谈到 OSSIM 中有大量准实时的统计数据，实际上

就是基于时间段的统计数据。这种数据不会实时更新，也很少需要增量更新，只有当达到重新 Build 该统计数据的时候需要做一次全量更新操作。虽然这种数据即使通过数据库来读取效率可能也会比较高，但是执行频率很高之后，同样会消耗不少资源。既然数据库服务器的资源非常珍贵，系统设计为放在应用相关的内存 Cache 中。

（2）在 Web UI 的仪表盘和 SIEM 控制台有大量个性化定制的数据，它们从访问频率来看相对于系统整体来说，也占了很大的比例，利用 Cache 可以加速访问。

（3）系统用户（特别是管理员）的基本信息，包括 session，用户的基本信息在应用系统中的访问频率极其频繁。所以用户基本信息的 Cache，很容易让整个应用系统的性能出现一个质的提升。

（4）OSSIM 各种配置及规则数据。由于这些配置信息变动的频率非常低，访问概率又很高，所以非常适合存放在 Cache 中，例如，现在要从不同的表里抽取数据以显示 Top 10，源码可参考/usr/share/ossim/www/panel/nids.php（在 25 行，TOP NID SEVENTS 描述中），由于 MySQL 对客户端每次提交的 SQL 不管是相同还是不同，都需要进行完全解析，这个动作主要消耗的资源是数据库主机的 CPU，SQL 语句的解析动作在整个 SQL 语句执行过程中整体消耗的 CPU 比例较多。

4. 检测 OSSIM 系统整体状态的命令行工具

在 OSSIM Server 和 Sensor 中都有一段基于 Python 的脚本，名为 alienvault_doctor，用它来检测系统状态。操作方法如下：

```
#alienvault_doctor
```

在 OSSIM Server 下的执行结果如图 5-16 所示。

```
     AlienVault version:                          4.8.0
     Installed profiles:        Server,Sensor,Framework,Database
     Operating system:                            Linux
     Hardware platform:                           x86_64
     Hostname:                                alienvault

Hmmm, let the Doctor have a look at you[Warning] Could not evaluate " "Task celerymethods.tasks.syst
em_tasks.sync_databases[3283805f-4c6c-4fa8-a33f-c79aaf6ce7ba] raised exception: UnboundLocalError("l
ocal variable 'rc' referenced before assignment",)" =="" in check "Celery workers": invalid syntax
(<string>, line 1)
..._
```

图 5-16　通过 alienvault_doctor 命令显示 Server 系统组件

在 Sensor 下的执行结果，如图 5-17 所示。

```
alienvault:/var/ossec/bin# alienvault-doctor

AlienVault Doctor version 4.15.1 (Mewes)

     AlienVault version:                      4.15.1-FREE
     Software profiles:                            Sensor

Hmmm, let the Doctor have a look at you..
```

图 5-17　通过 alienvault_doctor 命令显示 Sensor 系统组件

调用这一脚本的配置文件在/etc/ossim/doctor/目录下，执行检测脚本在 plugins 目录中，大家可深入研究一下这个脚本。通过该脚本的执行，我们可以轻易查看当前的终端是 OSSIM Server 还是 Sensor。

该命令可以显示 OSSIM 系统各组件的状态信息。可以检测系统磁盘，网络接口，服务器日志，数据库的状态等。该检测功能主要是调用/usr/share/alienvault/doctor/目录下的.pyc 文件（.pyc 是 python 编译后的二进制文件）。

5. 图形化监控工具软件

接下来为大家介绍几款图形化监控工具。

（1）nmon 工具可以为 Linux 提供监视和分析性能数据的功能，能查看的数据包括：

- CPU 使用率
- 内存使用情况
- 内核统计信息和运行队列信息
- 磁盘 I/O 速度、传输和读/写比率
- 文件系统中的可用空间
- 磁盘适配器
- 网络 I/O 速度、传输和读/写比率
- 页面空间
- 消耗资源最多的进程

Nmon 包括字符界面和图形化界面两种，字符界面如图 5-18 左边所示，安装方法如下：

```
#apt-get install nmon
```

图 5-18　nmon 和 xosview

与 nmon 功能相似的工具还有 Glances，它也是在命令行下执行的监视工具，其安装方法：

```
#apt-get update
#apt-get install python-pip build-essential python-dev
#pip install glances
```

（2）xosview 是另一款 Linux 下系统监测工具，它能显示如系统 CPU 使用率、平均负载、内存使用、交换空间、网络使用情况。如图 5-18 右边所示，安装方法如下：

```
#apt-get install xosview          \\前提是必须由 X-window 环境支持。
```

xosview 显示了诸如 CPU、负载、内存、交换分区和网络使用情况的信息。如果有完整的 Gnome 环境，可以在 Applications→System tools→xosview 菜单中找到启动图标，命令行启动方法：

```
#xosview  +net +page -ints
```

在 xosview 中具有多个选项，最常用的是“+net”监视网卡流量、“+page”显示页面交换频率。

如果首次启动 xosview，系统提示缺少字体，请执行以下命令：

```
#apt-get install xfonts-base                  \\必选项
```

重启 Gnome 后 xosview 即可使用。

6. 监控 MySQL 利器 mytop

mytop 是 OSSIM 系统默认提供的一款监控数据库命令行工具，效果比 phpmyadmin 稍差，由于数据库权限设置，默认远程查看数据库功能被禁止，需要首先按以下方法修改。

（1）在 OSSIM Server 控制台下操作

首先输入 root 用户及密码，然后进入 AlienVault Setup 界面，我们选择“Jailbreak System”，再选<Yes>选项，进入完整的“#”提示符。

接下来我们需要知道 MySQL 的 root 用户密码。

```
#cat /etc/ossim/ossim_setup.conf|grep pass
```

该命令输出第二行红色字体显示的 10 位密码就是我们要找的内容。

（2）测试

我们试着输入如下命令：

```
#mytop  -u root  -p 密码  -h 127.0.0.1 -d alienvault
```

见到如图 5-19 所示的界面，表示操作成功。

```
MySQL on localhost (5.5.29-29.4)                        up 0+01:35:21 [22:26:11]
 Queries: 201.0    qps:     0 Slow:     0.0       Se/In/Up/De(%):    1508/00/00/00
              qps now:     0 Slow qps: 0.0  Threads:    11 (   2/    9) 00/00/00/00
 Key Efficiency: 95.7%  Bps in/out:   0.7/198.1   Now in/out:   8.3/ 2.2k

        Id      User          Host/IP         DB      Time    Cmd Query or State
        --      ----          -------         --      ----    --- --------------
      1910      root        localhost alienvault         0  Query show full processlist
        78      root        localhost alienvault        12  Sleep
      2249      root        localhost alienvault        26  Sleep
        96      root        localhost alienvault        29  Sleep
      2073      root        localhost alienvault       268  Sleep
      2072      root        localhost alienvault       269  Sleep
      2070      root        localhost alienvault       282  Sleep
      2071      root        localhost alienvault       282  Sleep
      2060      root        localhost alienvault       284  Sleep
      2059      root        localhost alienvault       286  Sleep
      2058      root        localhost alienvault       296  Sleep
         1 event_sch        localhost                 5718 Daemon Waiting on empty queue
```

图 5-19　用 mytop 监控 MySQL 资源

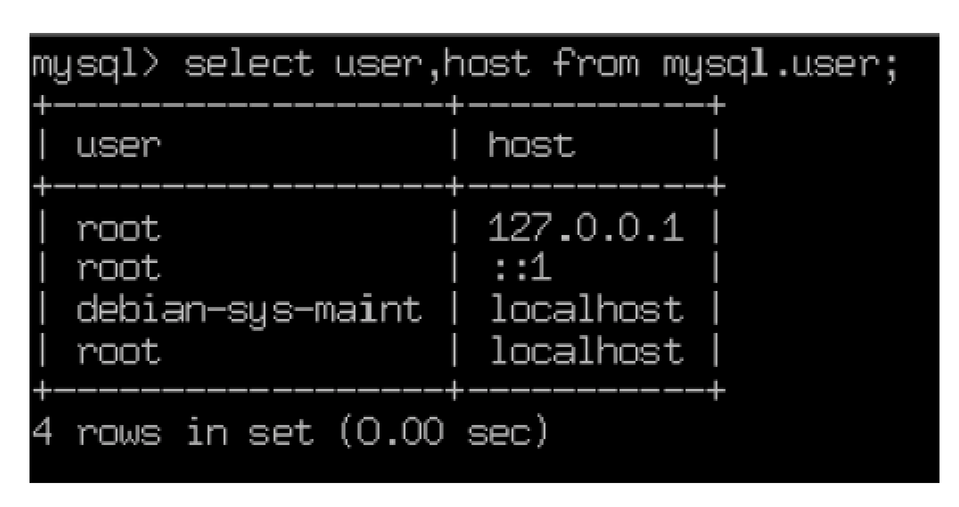
mytop 快捷键：

- s：设定更新时间
- p：暂停画面更新
- q：退出程序
- u：只查看某个使用者的线程
- o：反转所有行的排列顺序

如果你觉得系统默认的 root 命令很复杂，想自己指定一个 root 密码，可以用如下方法：

```
#ossim-db
```

在修改 root 密码之前，大家先查看系统中各用户的权限情况，如图 5-20 所示。



```
mysql> select user,host from mysql.user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| root             | 127.0.0.1 |
| root             | ::1       |
| debian-sys-maint | localhost |
| root             | localhost |
+------------------+-----------+
4 rows in set (0.00 sec)
```

图 5-20　查看 MySQL 权限

```
mysql>grant all privileges on *.* to 'root'@'localhost' identified by 'a1b2c3d4' with grant option;
mysql>flush privileges;
```

（3）通过 mytop 远程连接

工作中常常需要远程登录 MySQL 主机查看数据库，远程连接首要任务是解决数据库权限问题，假设我们需要让远程主机（这台主机的 IP 地址为 192.168.11.40）监控 OSSIM Server 中的 MySQL 数据库，我们还是在 OSSIM Server 端的命令行下输入“ossim-db”命令。

```
#ossim-db
```

接着进入 mysql>提示符下，继续操作。

```
mysql>grant all privileges on *.* to 'root'@'192.168.11.40' identified by 'a1b2c3d4' with grant option;
mysql>flush privileges;
```

有关权限设置的总结：MySQL 赋予用户权限可概括为：

```
grant 权限 on 数据库对象 to 用户@主机 identified by “密码”
```

修改权限以后，大家再用这条命令查看用户权限，如图 5-21 所示。

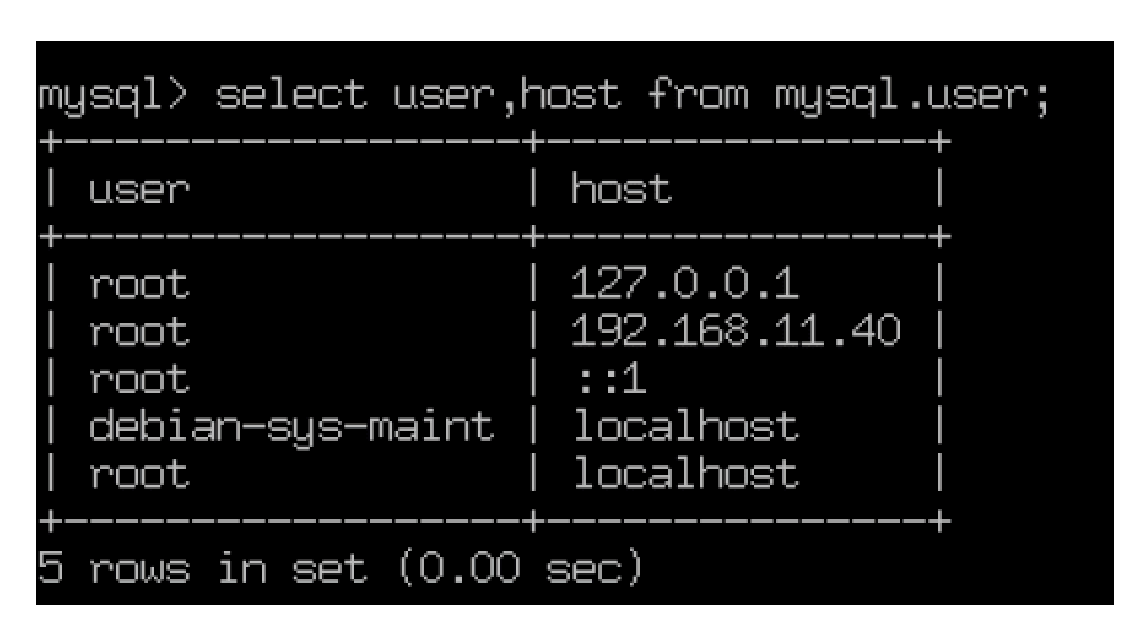

```
mysql> select user,host from mysql.user;
+------------------+---------------+
| user             | host          |
+------------------+---------------+
| root             | 127.0.0.1     |
| root             | 192.168.11.40 |
| root             | ::1           |
| debian-sys-maint | localhost     |
| root             | localhost     |
+------------------+---------------+
5 rows in set (0.00 sec)
```

图 5-21　检查修改权限

到目前为止，远端（192.168.11.40）主机可以用 mytop 监控到 OSSIM Server 中 MySQL 数据库吗？别忘了还有 iptables 服务。

（4）使用技巧

如果想经常监控某个数据库，不用每次都输入很多参数，可以将这些经常输入的参数先输入到文件，然后 mytop 会自动读取。

例如当前目录为/root，首先在当前目录下新建“.mytop”文件，内容如下：

```
#vi .mytop
user=root
pass=fHCtNi4dKN          \\上面获取的密码，也可以自己指定
host=127.0.0.1           \\因为这里我们在 OSSIM Server 本机上实验
db=alienvault
delay=10
port=3306
resolve=0
```

保存退出后，在命令行下直接输入 mytop 指令就能监控 alienvault 的状态了。

（5）危险的设置

有些读者为了访问方便而将数据库权限大门敞开，这样做非常危险。将第 3 步中 192.168.11.40 这个主机 IP 用“%”代替，表示任何主机都可以连接数据库，操作如下：

```
mysql>grant all privileges on *.* to 'root'@'%' identified by 'a1b2c3d4' with grant option;
```

（6）故障排除实例

使用 ossim-db 时出现“Access denied for user 'root'@'localhost'（using password:NO）”提示，如何处理？在图 5-22 中给出了此类故障排除的步骤。

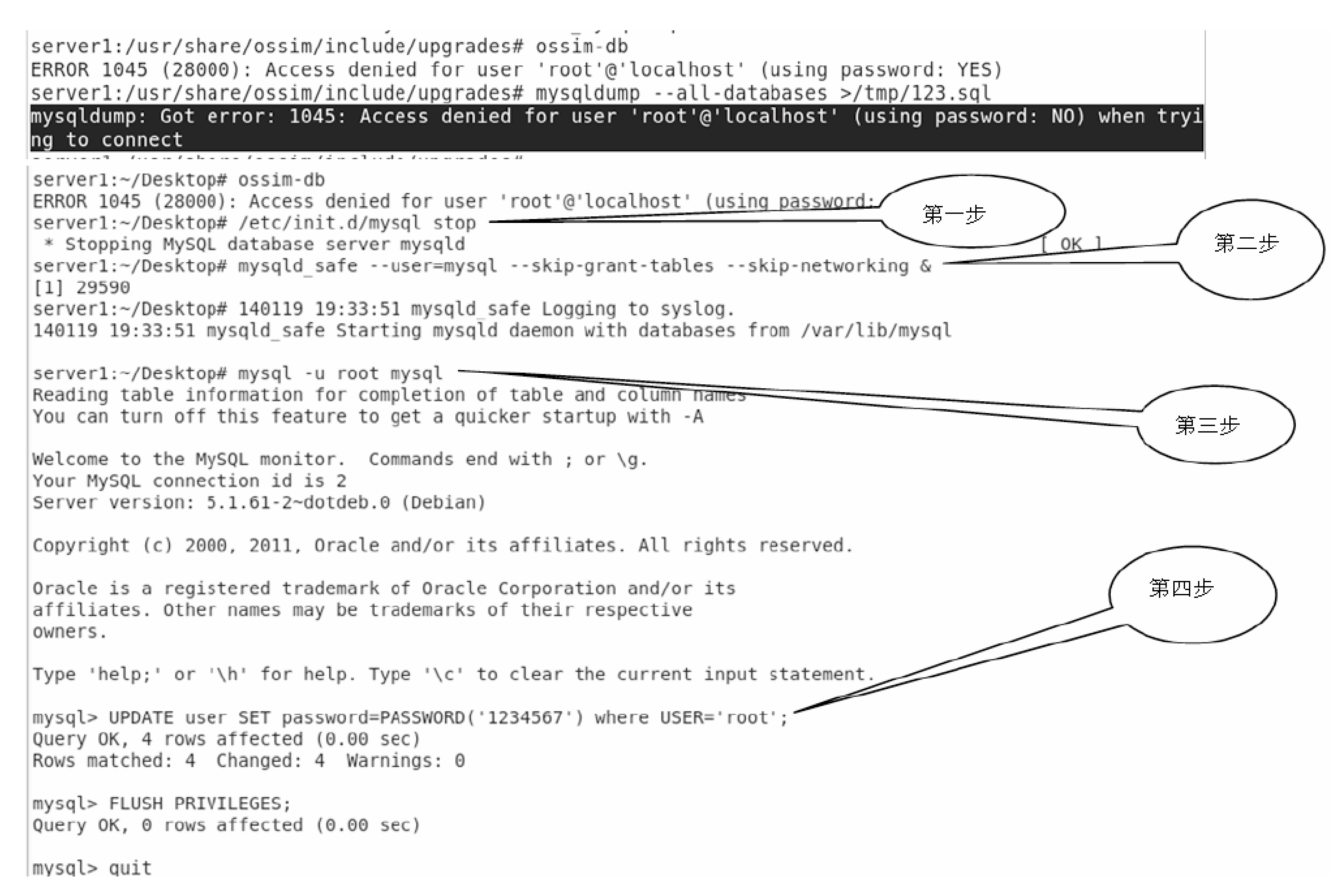


图 5-22　设置 MySQL 权限

7. 监控 Linux 系统资源和进程的工具

atop 是一款用于监控 Linux 系统资源与进程的工具，它以一定的频率记录系统的运行状态，所采集的数据包含系统资源（CPU、内存、磁盘和网络）使用情况和进程运行情况，并能以日志文件的方式保存在磁盘中。使用方法如下：

```
#atop
```

8. 如何找出最消耗内存的进程（smem）

第一种方法我们使用 ps 命令查看。

```
# ps auxw|head -1;ps auxw|sort -rn -k4|head -10
```

第二种使用 smem 工具，它的安装命令为 apt-get install smem。用法为：

```
# smem
```

9. 如何对 OSSIM 系统目录大小进行排序

我们知道使用 du 命令查看磁盘目录及文件的使用情况，这里介绍一款磁盘目录占用空间

情况的工具 ncdu。ncdu 命令是对传统 du 命令功能上的增强，不需要像 du 那样输入大量的命令（和它功能相当的命令有:du -sh /*），就可以计算文件及目录大小并可以按照文件大小或文件名进行排序。它有点类似于 Gnome 环境下的 Disk Usage Analyzer。

安装：apt-get install ncdu，使用："#ncdu / "，运行效果如图 5-23 所示。

```
ncdu 1.6 ~ Use the arrow keys to navigate, press ? for help
--- / ---------------------------------------------------------------------
    2.2GiB  /usr
    1.4GiB  /var
  110.9MiB  /lib
   76.3MiB  /etc
   18.8MiB  /boot
    5.8MiB  /bin
    4.3MiB  /sbin
    1.3MiB  /root
  164.0kiB  /dev
   56.0kiB  /tmp
   44.0kiB  /media
e  16.0kiB  /lost+found
    8.0kiB  /home
e   4.0kiB  /mnt
e   4.0kiB  /opt
e   4.0kiB  /selinux
e   4.0kiB  /srv
    0.0  B  /proc
    0.0  B  /sys
@   0.0  B   initrd.img
@   0.0  B   vmlinuz
 Total disk usage:   3.8GiB  Apparent size: 128.0TiB  Items: 262091
```

图 5-23 用 ncdu 列目录

常用快捷键：

- n：按文件名进行排序。
- s：按文件大小进行排序。
- r：重新统计当前文件夹大小。
- g：用#或百分比显示各文件/目录的大小所占的百分比。

从这个工具我们能方便地分析出，初始情况下的 OSSIM 系统，/var/、/usr 目录容量占据整个系统空间的 80%，为我们手动分区时提供了参考依据。

10. OSSIM 的流量监控工具 iftop

OSSIM 系统中可以使用 top 查看系统资源、进程、内存占用等信息，查看网络状态可以使用 netstat、nmap 等工具。若要查看实时的网络流量，监控 TCP/IP 连接等，则可以使用 iftop 命令，其运行情况如图 5-24 所示。在 OSSIM 中 iftop 安装方法为：

```
#apt-get install iftop
#iftop
```

```
              1.91Mb            3.81Mb            5.72Mb            7.63Mb       9.54Mb
169.115.209.139.adsl-pool.jlccptt.n  => localhost          5.24Kb  5.24Kb  5.41Kb
                                     <=                     122Kb   107Kb   103Kb
58.243.167.124.adsl-pool.sx.cn       => localhost          7.98Kb  5.86Kb  5.68Kb
                                     <=                     138Kb   103Kb   105Kb
123.123.224.243                      => localhost          2.84Kb  5.18Kb  5.44Kb
                                     <=                    68.4Kb  96.8Kb   100Kb
51.218.163.124.adsl-pool.sx.cn       => localhost          5.52Kb  4.90Kb  4.62Kb
                                     <=                    85.6Kb  96.5Kb  97.3Kb
localhost                            => 61.49.240.1          480b    480b    480b
                                     <=                      480b    480b    480b
localhost                            => localhost            424b    522b    522b
                                     <=                      312b    365b    389b
localhost                            => localhost              0b    522b    370b
                                     <=                        0b    365b    274b
101.47.75.251                        => localhost              0b      0b      0b
                                     <=                        0b    323b    324b
221.3.132.205                        => localhost              0b      0b      0b
                                     <=                        0b    322b    324b
alienvault.alienvault                => localhost              0b    114b    511b
                                     <=                        0b    192b    919b
101.66.214.180                       => localhost            260b     52b     42b
                                     <=                      560b    224b     97b
localhost                            => localhost              0b     85b    130b
                                     <=                        0b     62b     91b
114.250.171.212                      => localhost              0b     64b    103b
                                     <=                        0b     64b    556b
222.130.22.149                       => localhost              0b      0b      0b
                                     <=                        0b    112b     83b
121.29.11.126                        => localhost              0b      0b     10b
                                     <=                        0b    110b     83b
218-161-9-221.HINET-IP.hinet.net     => localhost              0b     45b     11b
                                     <=                        0b     53b     13b
60.31.30.132                         => localhost            200b     40b     68b
                                     <=                      212b     42b     68b
──────────────────────────────────────────────────────────────────────────────────
TX:            cumm:  46.0KB   peak:   1.42Kb          rates:    0b    114b    511b
RX:                    117MB            493Kb                439Kb   429Kb   445Kb
TOTAL:                 117MB            494Kb                439Kb   430Kb   446Kb
```

图 5-24　iftop 运行情况

界面上面显示类似刻度尺。中间的“<=”、“=>”这两个左右箭头，表示流量方向。iftop 界面上一些栏目信息说明如下。

- TX：发送流量。
- RX：接收流量。
- TOTAL：总流量。
- cumm：从运行 iftop 起到目前的总流量。
- peak：流量峰值。
- rates：分别表示过去 2s、10s、40s 的平均流量。

常用的参数：

- -i：设定监测的网卡，如#iftop -i eth1。
- -n：使 host 信息默认直接都显示 IP，如#iftop -n。
- -N：使端口信息默认直接都显示端口号，如# iftop -N。
- -F：显示特定网段的进出流量，如#iftop -F 10.10.1.0/24，或# iftop -F 10.10.1.0/255.255.255.0。
- -h：显示帮助信息。
- -p：使用该参数后，中间的列表显示的本地主机信息，出现了本机以外的 IP 信息。
- -b：使流量图形条默认显示。
- -f：过滤计算包。
- -P：使主机信息及端口信息默认显示。

- -m，设置界面最上边的刻度的最大值，刻度分五段显示，如# iftop -m 100M。

11. 如何利用 Apache 自带工具 ab 测试 OSSIM 响应速度

ab（apache benchmark）是 Apache 自带的性能测试工具，下面我们用 ab 来测试 OSSIM 首页，操作效果如图 5-25 所示。

```
alienvaulttest:~# ab -A auth-admin:1234567 -c 100 -n 100 https://192.168.150.114/ossim/session/login
.php
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.150.114 (be patient).....done


Server Software:        Apache
Server Hostname:        192.168.150.114
Server Port:            443
SSL/TLS Protocol:       TLSv1/SSLv3,DHE-RSA-AES256-SHA,2048,256

Document Path:          /ossim/session/login.php
Document Length:        9932 bytes

Concurrency Level:      100
Time taken for tests:   3.355 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1026600 bytes
HTML transferred:       993200 bytes
Requests per second:    29.81 [#/sec] (mean)
Time per request:       3354.691 [ms] (mean)
Time per request:       33.547 [ms] (mean, across all concurrent requests)
Transfer rate:          298.85 [Kbytes/sec] received
```

图 5-25　利用 ab 对 OSSIM Web 进行测试

```
#ab -A auth-admin:1234567 -c 100 -n 100
https://192.168.120.78/ossim/session/login.php
```

参数解释：

- -A：认证用户名和密码。
- -n：在测试会话中所执行的请求个数。默认时，仅执行一个请求。
- -c：一次产生的请求个数。默认是一次一个。

这个表示同时处理 100 个请求并运行 100 次 login.php 文件，Web UI 用户名和密码分别是 admin、1234567。

12. 如何详细了解 OSSIM 系统进程的网络带宽占用情况

在 OSSIM 系统性能调优的时候，经常需要知道系统中每个进程占用带宽的情况，OSSIM 下安装非常方便：

```
#apt-get install nethogs        \\*安装 nethogs
#nethogs eth0                   \\*使用 nethogs
```

nethogs 最新版本为 0.8.0，有别于其他流量监控工具（例如 iftop），nethogs 可以细化到显示系统中每个进程的带宽占用情况，这样就能更直观地获取网络使用情况，例如查看到底是哪个进程占用了大量的带宽，或者是哪个 IP 对服务器进行了访问。它支持 IPv4 和 IPv6 协议、支持本地网卡及 PPP 连接。

除此之外，在 OSSIM 下还可以通过上面的方式安装诸如 iftop、bwm-ng、ipfm、speedometer、pktstat、nload、vnstat、iptraf、ifstat、mtr、nethogs 等命令行下的网卡流量监控工具。

13. 为 OSSIM 系统进行压力测试 tcpreplay

这里以 tcpreplay 工具为例讲解如何进行测试，例如，当前目录下有 123.pcap 抓取的数据包文件，重放 aurora.pcap 包，如图 5-26 所示。

```
alienvault:~# tcpreplay --intf1=eth0 aurora.pcap
sending out eth0
processing file: aurora.pcap
^C
Actual: 25 packets (14120 bytes) sent in 2.22 seconds
Rated: 6360.4 bps, 0.05 Mbps, 11.26 pps
alienvault:~#
```

图 5-26　重放 aurora.pcap 包

```
#tcpreplay --intf1=eth0 123.pcap
```

tcpreplay 会用前面抓到的以 libpcap 格式存储的数据包来测试各种网络设备，包括重新回放 Demo Virus 包，有关测试用 worm pcap 数据包，可到作者博客下载。

--intf1=eth0 是指主接口是 eth0，客户机→服务器的数据包通过此接口发出。

```
#tcpreplay -l 10 -p 1000 -i eth0 123.pcap
```

-l 参数代表循环多少次，-p 参数代表每秒发多少个包，-i 代表从那个网卡发出。

```
#tcpreplay -l 10 -p 1000 -i eth0 *.pcap     \\发送当前目录下所有 pcap 包
#tcpreplay -t -i eth0 *.pcap     \\-t 参数表示全速--topspeed，全速回放当前目录下所有 pcap 包。
```

如果目录下有多个 pcap 文件需要合并成一个 pcap 文件，可以使用如下命令：

```
#mergecap -w output.pcap  input1.pcap input2.pcap
```

需要说明，对于典型数据包最为运维人员需要了解，可以到网址 https://wiki.wireshark.org/SampleCaptures/，或者 http://www.netresec.com/?page=PcapFiles 下载 Demo 数据包，在本书最后一章中，将会为大家讲解利用一些典型的蠕虫病毒爆发时的数据包进行分析的方法。

当然你也可以通过 suricata 读入 pcap 包，方法如下：

```
alienvault:~#suricata -c /etc/suricata/suricata.yaml -r tmp3.pcap
```

- -c：这个选项是最重要的选项。在-c 之后，要输入 suricata.yaml 文件所在的路径。

- -r：在这个选项之后，你可以输入记录数据的抓包文件的路径和文件名。可以在 pcap/offline 模式下，在这个文件中查看包数据。

14. 压力测试工具 Tsung

Tsung 是一个非常好用的压力测试软件，它基于 Erlang 语言，支持多种协议，如 xmpp、http 等。使用它可以测试 OSSIM 中消息服务器能承受多大压力。在 OSSIM 系统中安装 Tsung 非常容易，其安装效果如图 5-27 所示。

```
alienvault:~/tsung-1.5.1/src/tsung/tsung# wget http://tsung.erlang-projects.org/dist/debian/tsung_1.
5.1-1_all.deb
--2014-08-06 13:43:21--  http://tsung.erlang-projects.org/dist/debian/tsung_1.5.1-1_all.deb
Resolving tsung.erlang-projects.org... 176.31.254.13
Connecting to tsung.erlang-projects.org|176.31.254.13|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1133184 (1.1M) [application/x-debian-package]
Saving to: `tsung_1.5.1-1_all.deb'

100%[==========================================================>] 1,133,184   30.8K/s   in 43s

2014-08-06 13:44:07 (25.5 KB/s) - `tsung_1.5.1-1_all.deb' saved [1133184/1133184]

alienvault:~/tsung-1.5.1/src/tsung/tsung# dpkg -i tsung_1.5.1-1_all.deb
Selecting previously deselected package tsung.
(Reading database ... 61451 files and directories currently installed.)
Unpacking tsung (from tsung_1.5.1-1_all.deb) ...
Setting up tsung (1.5.1-1) ...
Processing triggers for man-db ...
```

图 5-27　Tsung 安装和测试效果

将 tsung_1.5.1_1_all.deb 安装完成后，开始编写 tsung.xml 配置文件，其实不用你写，只要将/usr/share/doc/tsung/examples 目录下的实例根据需要进行少量修改即可使用。

开始运行 Tsung ：

```
#tsung -f youfile.xml start
```

程序运行后数据保存在~/.tsung/log 目录下。

15. hping3 使用的实例

大家知道 hping3 是检测服务器防御性能的工具，它不但能测试出服务器所承受的网络压力，而且还能对网络中的端口进行扫描，同时它还是一种 DDoS 软件，能够产生 DDoS 攻击，因而用户在使用的过程中不得用于非法用途。

在 OSSIM 系统中安装 hping3 的方法：

```
#apt-get install hping3
```

安装完毕之后，下面看几个实例：

（1）使用 hping3 进行 SYN Flood 模拟：

```
#hping3 -q -n -a 10.0.0.1  -S  -s 53 --keep -p 22 --flood 192.168.0.2
```

参数说明：

- -a：参数可以进行源 IP 地址的伪造。
- -s：表示源端口。
- -S：表示设置 SYN 标识。
- -p：22 表示设置数据包的目标端口为 22。

10.0.0.1 为源地址、192.168.0.2 为目标地址，停止 hping3 程序按 Ctrl+C。

（2）SYN flood 测试：

```
#hping3 -q -n -a 10.0.0.1  -S  -s 53 --keep -p 22 --flood 192.168.0.2
```

10.0.0.1 为源地址，192.168.0.2 为目标地址。

构造源地址为 192.168.10.99，并使用 1000 微秒的间隔发送各个 SYN 包。

```
#hping3 -I eth0 -a 192.168.10.99 -S 192.168.10.33 -p 80 -i u1000
```

参数说明：

- -I eth0：指定使用 eth0 端口。
- -c –count：指定收发数据包的个数。
- -i –interval：指定发包间隔为多少毫秒，如-i m10 表示发包间隔为 10 毫秒、-i u1 表示每个数据包发送间隔为 1 微妙。

（3）进行 SARFU 扫描的例子：

```
#hping3 -q -n -a 10.0.0.1  -SARFU -p 22 --flood 192.168.0.2
```

（4）利用 hping3 进行 UDP Flood 的例子：

```
#hping3 -q -n -a 10.0.0.1  --udp  -s 53 --keep -p 68 --flood 192.168.0.2
```

（5）ICMP flood 的例子：

```
#hping3 -q -n -a 10.0.0.1 --id 0  --icmp -d 56 --flood 192.168.0.2
```

16. OSSIM 的 TCP 扫描工具

（1）Knocker

Knocker 是一个简单易用的 TCP 端口扫描工具，采用 C 语言编写，用于分析主机上运行的服务。

安装：

```
#apt-get install libio-socket-ssl-perl
#apt-get install knocker
```

使用：

```
#knocker -H 192.168.150.10 -SP 1 -EP 10000
```

- -H：目标主机。
- -SP：开始端口。
- -EP：结束端口。

（2）sendip

由于在测试 IDS 中经常需要构造各种 TCP、UDP 和不同大小的包，sendip 为我们提供该功能，其运行效果如图 5-28 所示，安装方法如下：

```
#apt-get install sendip
#sendip -v -p ipv4 -is 192.168.120.78 -id 192.168.120.78 -p udp -us 8000 -ud 4000 192.168.120.78 -d dfsdfasdf
```

```
alienvault:/tmp# sendip -v -p ipv4 -is 192.168.120.78 -id 192.168.120.78 -p udp -us 8000 -ud 4000 19
2.168.120.78 -d dfsdfasdf
Added 26 options
Initializing module ipv4
Initializing module udp
Finalizing module udp
Finalizing module ipv4
Final packet data:
45 00 00 25   E..%
DA 02 00 00   ....
FF 11 6F D7   ..o.
C0 A8 78 4E   ..xN
C0 A8 78 4E   ..xN
1F 40 0F A0   .@..
00 11 47 6D   ..Gm
64 66 73 64   dfsd
66 61 73 64   fasd
66   f
Sent 37 bytes to 192.168.120.78
Freeing module ipv4
Freeing module udp
alienvault:/tmp#
```

图 5-28　sendip 运行效果

上面结果说明：发送源地址为 192.168.120.78，源端口为 8000 的 udp 包，目的地址为 192.168.120.78，目的端口为 4000 设备上，数据包的内容是“dfsdfasdf”，如果要直接引用一个文件的内容的话，可以使用-f 参数，后面写文件的路径，例如：

```
# sendip -v -p ipv4 -is 192.168.96.7 -id 192.168.150.115 -p udp -us 8000 -ud 4000 192.168.150.115 -f /tmp/123.log
```

举例 1：

下面这段 Shell 脚本很容易，脚本内容如图 5-29 所示。

```
#! /usr/bin/perl -w
$i=1;
  while ($i<2)
  {
for ($j=0;$j<=65;++$j)
{
system"sendip -v -d r1573 -p ipv4 -iv 4 -ih 5 -il 1520 -ifm 1 -if 0 -iy 8 -is 192.168.150.115 -id 19
2.168.150.115 -p udp -us $j -ul 56 192.168.150.115"
}
++$i
}
```

图 5-29　一段包含 sendip 的脚本

sendip 可以发送 NTP、BGP、RIP、TCP、UDP、ICMP、IPv4 和 IPv6 等各种格式的数据包，sendip 本身是以模块的方式发送各种协议的数据包，用“-p”参数指定协议类型，要发送每种协议的数据包，必须对该协议的数据包格式有一定的了解。通常发送 TCP/UDP/ICMP 数据包时，都必须以 IP 包进行封装，然后才可以发出去。

举例 2：

```
#sendip -v -d r64 -p ipv4 -iv 4 -ih 5 -il 128 -is 10.0.0.1 -id 10.0.0.2 -p tcp
-ts 1379 -td 23 -tt 8 10.0.0.2
```

- -v：运行时输出详细运行信息，如不指定，运行时将不输出信息。
- -d r64：用 64 字节的随机数值填充 IP 包中的数据段。
- -p ipv4：指定协议类型为 IP 协议（IP 协议有自己的相应参数，以 i 开头）。
- -iv 4：协议版本为 4，即 IPV4。
- -ih 5：指定 IP 头的长度为 5 × 4 = 20 字节。
- -il 128：指定 IP 包的总长度为 128 字节。
- -is 10.0.0.1：指定 IP 包的源地址。
- -id 10.0.0.2：指定 IP 包的目的地址。
- -p tcp：指定 IP 包中封装的包的协议类型（TCP 协议有自己的相应参数，以 t 开头）
- -ts 1379：指定 TCP 包的源端口 1379
- -td 23：指定 TCP 包的目的端口为 23
- -tt 8：指定 TCP 包的偏移量即 TCP 头的长度，没有 TCP 选项时为 5，即 20 字节，有 TCP 选项时需要增加。
- 10.0.0.2：指定发包的目的主机。

类似这种发包工具很多，比如数据包发送工具 Packet Sender、ostinato，也可用来作为发包的工具，使它能够发送和接收通过 TCP 和 UDP 的网络数据包，下载地址为 http://packetsender.com/。

5.7 小结

本章从 Linux 的性能优化技巧开始讲起，逐步过渡到故障排除，最后到 OSSIM 系统的优化方案，并运用各种性能监测工具来发现 OSSIM 系统出现的性能瓶颈。

第 6 章

Snort 规则分析

从本章可以学习到：

- 预处理器
- Snort 日志分析
- 数据包记录模式
- 规则分析与编写
- 可疑流量监测
- 异常行为分析

本章将详细讲解 OSSIM 平台下的一个重要组成部分 Snort，它是一种拥有实时（Real-Time）流量分析、网络 IP 数据包（Packet）记录等特性的强大的网络入侵检测/防御系统。由于 OSSIM 已完美集成 Snort 系统，所以本章讲解会集中在 Snort 的应用场景、应用方法和规则分析上，这些内容也是 Snort 真正的内涵所在。本章讲解 Snort 的内容是以后使用 Suricata 的基础，如不特别指明，所有实验环境主要以 OSSIM 3 为基础。

6.1 预处理程序

6.1.1 预处理器介绍

预处理器的作用是通过一些规则的设置来提前检测数据包是否存在入侵的风险。所有预处理器的输入和输出都是报文，但每个预处理器对报文结构的字段处理各有不同。该模块由功能各不相同的预处理器组成，可实现的功能包括分片重组、端口扫描检测、系统性能监测等。一些预处理器可以通过对数据包头部进行检测，如果发生异常则产生告警，这为 Snort 检测引擎分析数据包打下了基础。以 Snort 2.9.0.5 版本为例提供了 14 个预处理器插件。表 6-1 对它们的功能做了简单的描述。

表 6-1　预处理程序分类及功能

插件分类	功　能
Frag3	重组 IP 分片，只处理属于 IP 分片的数据包，并对一些基于 IP 分片的攻击进行检测
Stream5	可对 TCP、UDP 等报文进行会话重组，跟踪 TCP、UDP、ICMP 会话的状态信息，可检测出一些针对 TCP 协议的攻击等
ARP Spoof	针对 ARP 报文的某些类型检测
BO	检测 Back Office 攻击
HttpInspect	该插件解析 HTTP 报文，发现一些针对 HTTP 协议的攻击，只能一次检测一个报文，而不能将一个 HTTP 会话中不同的报文关联起来进行检测
Performance Monitor	通过多种统计指标监测 Snort 运行期间的性能，并汇总到指定位置
sfPortscan	检测端口扫描行为
SMTP	检测 SMTP 流量，并对数据内容进行规范化
Ftp/Telnet	检测 FTP/TELNET 流量，并对数据内容进行规范化
SSH	检测 Challenge-Response 缓冲溢出攻击、CRC32（循环冗余校验）攻击
SSL/TLS	检测 SSL/TLS 流量
DNS	处理 DNS 应答，可检测 DNS Client Data 溢出，废弃的记录类型

在配置 snort.conf 时，有下列预处理程序大家必须了解。

1. Frag3

frag3 预处理能检测与 IP 包分段有关的攻击类型，它是 Snort 应对 IP 分段攻击的有效武器。该预处理程序对于 Snort 非常重要，因此不要禁用它。

自 Snort 2.7.0 版起，采用了 frag3 代替原先的 frag2，还加入了新的元素 stream5。frag3 数据处理能力更强，而且适用于分布式环境。

为了更好地认识它，我们首先打开/etc/snort/snort.conf 文件，查看 197 行内容，如图 6-1 所示：

```
197 # Target-based IP defragmentation.  For more inforation, see README.frag3
198 preprocessor frag3_global: max_frags 65536
199 preprocessor frag3_engine: policy first detect_anomalies overlap_limit 10 min_fragment_length 10
   0 timeout 180
```

图 6-1　frag3 参数配置

我们知道，以太网中任何一个长度超过 1500 bytes 的 IP 数据包必须进行分段，这种分段，在源端和中介路由器都会发生，在数据分段后，在目标主机上重装，攻击者可以利用这一重新装配过程进行攻击。这里，攻击者就是利用分段重写 TCP 头数据来绕过防火墙和 IDS 设备的。

为了方便实验，大家不必从头安装这个工具，可以在 BackTrack5 工具箱中直接找到 fragroute 工具，用来模拟这种攻击。我们可以通过 frag3 的一些附加的配置选项，检测这种类型的攻击。

接下来我们讲解 frag3 选项，该选项包括“preprocessor frag3_global”和“preprocessor frag3_engine”两个子项，每一项都有参数描述。下面讲解一下主要参数。

（1）timeout（单位为秒，用 s 表示）

设置分段可以持续的秒数，如果分段没有在规定的时间内完成，就会被丢弃，默认值是 60s。如果被监控网段经常出现 IP 分段，例如有大量的分段流量，建议减少该数值，以避免误报。假如攻击者猜测到 frag3 默认值为 60s，就会利用该信息来躲避监控，这时可以适当增大该值来防止这种攻击。注意：在 OSSIM 里设置为 180 s。

（2）memcap

memcap 表示 frag3 预处理可以使用的内存大小，默认为 4MB。

（3）min_ttl

指定 frag3 能接收包的最小 TTL 值，默认值为 1，取值范围是 1~255。只要低于该值就会被丢弃。我们知道对于数据包来讲，TTL 就好比一个“死亡”计数器，被路由器传递一次就减 1，如果 TTL 值到达 0，该包就会被丢弃，不再继续传递。攻击者利用这一特性，在攻击包中插入垃圾数据，就能使 Snort 无法识别，从而不能检测出这种攻击。例如，假设图 6-1 有 3 个包是针对 FTP Server 的攻击：

Packet 1
QUOTESITE　　TTL=60

Packet 2
inserted datadata TTL=1

Packet 3
EXEC echo toor::0:0:/:/bin/sh>>/etc/passwd TTL=120

Snort 在收到这三个数据包后会将该攻击看成如下内容：

QUOTE SITE	inserted datadata	EXEC echo toor::0:0:/:/bin/sh>>/etc/passwd

正是由于插入了第 2 个包的垃圾数据“inserted datadata”，所以 Snort 没有任何一条规则和它匹配，而垃圾数据在到达目标主机之前，由于耗尽 TTL 而被丢弃，所以目标主机接收到的真实攻击为：

QUOTE SITE	EXEC echo toor::0:0:/:/bin/sh>>/etc/passwd

这样精心处理的数据包到达对方主机就有可能成功实施攻击。所以我们在该选项中合理设定这个值，以丢弃那些 TTL 值很低的包，这样就可以防止这种躲避技术。

（4）overlap_limit

检测每个包的重叠分片数量，默认为 0（代表不限制）。如果希望进行异常检测就必须选中它。由图 6-1 可以看出在 OSSIM 系统里其值设置为 10。

（5）min_fragment_length

min_fragment_length 定义了最小分片大小（即有效载荷的大小）在进行异常检测时，如果数据包的分段小于等于这个限定值，就会被认为是恶意行为。min_fragment_length 默认为 0，在系统中该值设定为 100。

（6）policy

选择目标的碎片整理模式，可用模式有：first、last、bsd、linux、windows 和 solaris。默认为 bsd 模式。

2. Stream5

Stream5 和上面介绍的 frag3 有点类似，它可预处理被用来维持 TCP 流的状态，检测某些信息收集类型的攻击。Stream5 预处理器使 Snort 具备流重组和状态分析的能力，能够跟踪 TCP 和 UDP 会话。Stream5 给用户提供超过 256 个 TCP 同步连接，最大能够处理 1,048,576 个 TCP 同步连接。Stream 5 还可以检测和处理数据覆盖、TCP 时间戳、SYN 包存在净荷、FIN 和 RESET 序列号等方面存在的异常，表 6-2 对 stream5 的主要选项进行了描述。

表 6-2　Stream5 选项及描述

选 项	描 述
track tcp <yes|no>	TCP 会话跟踪，默认为启用
max tcp <num sessions>	最大同时 TCP 会话跟踪，默认为 262144，取值范围是 1~1048576
memcap <num bytes>	内存容量选项，用于限制 Stream5 预处理可以使用的内存大小，默认为 8388608（8MB），范围是 32768~1073741824，也就是 32KB~1GB，如果监控网段有大量会话，应该谨慎增加这一值
max ip <num sessions>	最大同时 IP 会话跟踪数，默认是 16384，范围是 1~1048576
track udp <yes|no>	默认启用 UDP 跟踪会话
track icmp <yes|no>	默认关闭 ICMP 跟踪会话
timeout	设置对话可以持续的秒数，如果对话没有在规定的时间内完成，就会被清理。默认值为 30s，系统里设定为 180s
detect anomalies	表示启用 TCP 协议异常检测报警

利用 Stream5 进行状态检查，可以帮助 Snort 匹配多种攻击。例如 Stream5 预处理能检测

到攻击者利用 TCP 流隐藏信息收集流量的试探，最有代表的就是 TCP 半开放式扫描，利用这种扫描方式，TCP 三次握手不会完成，也就是攻击者不再会发送 ACK 包完成三次握手。该方式能骗过没有监控流状态的老式 IDS，它还可以进行不断扩展，用以对付一种引起针对 Snort 拒绝服务的无状态攻击。我们接着看配置文件，如图 6-2 所示。

```
# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: max_tcp 8192, track_tcp yes, track_udp yes, track_icmp no max_activ
e_responses 2 min_response_seconds 5
preprocessor stream5_tcp: policy first, detect_anomalies, require_3whs 180, \
   overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
    ports client 21 22 23 25 42 53 79 109 110 111 113 119 135 136 137 139 143 \
        161 445 513 514 587 593 691 1433 1521 2100 3306 6070 6665 6666 6667 6668 6669 \
        7000 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \
    ports both 80 311 443 465 563 591 593 636 901 989 992 993 994 995 1220 1414 1830 2301 2381 2
809 3128 3702 5250 6907 7001 7702 7777 7779 \
        7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914 7915 7916 \
        7917 7918 7919 7920 8000 8008 8028 8080 8088 8118 8123 8180 8243 8280 8888 9090 9091 944
3 9999 11371
preprocessor stream5_udp: timeout 180
```

图 6-2　Stream5 参数配置

该 Stream5 预处理器是基于针对 Snort 目标 TCP 重组模块，它完全取代原先 Snort 版本中的 Stream4 和流处理器，它能跟踪 TCP/UPD 会话。Stream5 全局配置格式如下：

```
preprocessor stream5_global: \
           [track_tcp <yes|no>], [max_tcp <number>], \
           [memcap <number bytes>], \
           [track_udp <yes|no>], [max_udp <number>], \
           [track_icmp <yes|no>], [max_icmp <number>], \
           [track_ip <yes|no>], [max_ip <number>], \
           [flush_on_alert], [show_rebuilt_packets], \
           [prune_log_max <bytes>], [disabled]
```

3. RPC decode

Snort 从 2.7.0 的版本起，采用 HTTP_decode 作为预处理，负责检测异常的 HTTP 流量，并且使其标准化，以便检测引擎能正确地对其解释。RPC_decode 预处理与 HTTP_decode 功能相似，不过它是针对 RPC 协议。RPC 能被攻击者用来侦察和远程攻击。攻击者利用端口映射程序，比如 rpcbind 和 portmapper 有可能对远程服务动态绑定，攻击者利用 rpcbind 收集信息、发现其他目标、进行缓冲区溢出攻击或者 RPC 服务攻击。

一些高明的攻击者希望通过隐藏RPC流量来躲避IDS系统的监控。如RPC特征“0186A0”，这一特征被分散在多个包内，那么 Snort 就无法对其匹配。所以我们要用 RPC_decode 预处理进行标准化。RPC_decode 有一个配置选项“ports list”，它可以列出所需要用 RPC_decode 进行标准化处理的 RPC 端口，其默认值为 111 和 32770~32779，在 OSSIM 中的/etc/snort/snort.conf 文件中，第 248 行，参数配置如图 6-3 所示。

```
# ONC-RPC normalization and anomaly detection.  For more information, see the Snort Manual, Conf
iguring Snort - Preprocessors - RPC Decode
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no_aler
t_multiple_requests no_alert_large_fragments no_alert_incomplete

# Back Orifice detection.
preprocessor bo
```

图 6-3　rpc decode 参数配置

4. BO

BO（Back Orifice）预处理检测，有代表的实例是古老的 BO2K（Back Orifice 2000），它除了可以用于 Winows95/98 系统，也可以用在 Windows NT/2000 操作系统上。BO2K 包含了 Back Orifice 所具有的功能。我们在 OSSIM 3 系统的/etc/snort/rules/backdoor.rules 规则文件中就能找到 Back Orifice 2000 backdoor。它虽然是古老的木马，但以它为原型，改进成更加先进的特洛伊木马，依然具有一定的威胁。一些脚本小子们常利用它获取远程系统的控制权。启用 BO 检测只需在 snort.conf 中加入下面一行，在 OSSIM 中配置如图 6-3 所示。

```
Preprocessor bo
```

5. FTP 和 Telnet

由于 Telnet 和 FTP 协议有关，在预处理中放到了一起。它可以对 Telnet 或 FTP 流中的任意嵌入二进制控制代码进行解码。黑客在试探中将控制代码插入流量中来躲避 Snort 检测。控制代码插入常利用 SITE EXEC 这个 FTP 命令也和它的漏洞有关。例如攻击者利用 FTP 命令 SITE EXEC，通过 FTP 链接来执行系统命令，键入如下所示的 FTP 命令就可以收到 passwd 文件：

```
QUOTE SITE EXEC echo toor::0:0::/:/bin/sh >>/etc/passwd
```

有些攻击者可以将 Telnet 控制代码分布在这条命令中，希望躲避 Snort 检测。有关 ftp_telnet 参数配置参看 OSSIM 配置文件/etc/snort/snort.conf 第 315 行，如图 6-4 所示。

```
# FTP / Telnet normalization and anomaly detection.  For more information, see README.ftptelnet
preprocessor ftp_telnet: global inspection_type stateful encrypted_traffic no
preprocessor ftp_telnet_protocol: telnet \
    ayt_attack_thresh 20 \
    normalize ports { 23 } \
    detect_anomalies
preprocessor ftp_telnet_protocol: ftp server default \
    def_max_param_len 100 \
    ports { 21 2100 3535 } \
    telnet_cmds yes \
    ignore_telnet_erase_cmds yes \
    ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
    ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
    ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
    ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
    ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
    ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
    ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
    ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
    ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
    ftp_cmds { XSEN XSHA1 XSHA256 } \
    alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP PASV PWD QUIT REIN STOU SYST XCUP XPWD }
 \
    alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR STOU XMKD } \
    alt_max_param_len 256 { CWD RNTO } \
    alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
    chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
    chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
    chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
    chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \
    chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \
    chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \
```

图 6-4　ftp_telnet 参数配置

6. ARPspoof

ARPspoof 是一个检测 ARP 协议流量的预处理程序。目前涉及 ARP 的攻击种类繁多，它们都是以 ARP 欺骗为基础，ARP 欺骗通过构造特殊 ARP 请求和应答包来达到目的，而伪造的 ARP 应答包，被存储在接收计算机的 ARP 缓存中。

ARP 欺骗的危害是能够误导流量，这很可能造成交换网络的一种试探，ARP 欺骗也能用来进行一些简单而有效的 Dos 攻击，当伪造的封包中包含无效的 IP-MAC 映射时，ARP 应答就会淹没目标设备 ARP 缓存，从而导致 DOS 攻击 ，这将引起目标设备向不存在的设备发送流量。而 ARPspoof 预处理，不但能检测此类攻击和 ARP 欺骗试探，而且能检测到 ARP 高速缓存重写攻击，它的配置选项为 host IP address host MAC address（主机 IP 地址主机 MAC 地址），参数配置在 snort.conf 的 400 行，如图 6-5 所示。

```
# ARP spoof detection.  For more information, see the Snort Manual - Configuring Snort - Preprocesso
rs - ARP Spoof Preprocessor
preprocessor arpspoof
preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

# SSH anomaly detection.  For more information, see README.ssh
preprocessor ssh: server_ports { 22 } \
                  autodetect \
                  max_client_bytes 19600 \
                  max_encrypted_packets 20 \
                  max_server_version_len 100 \
                  enable_respoverflow enable_ssh1crc32 \
                  enable_srvoverflow enable_protomismatch
```

图 6-5　ARPspoof 配置

图中的例子为：

```
192.168.40.1  f0:0f:00:f0:0f:00
```

若想用 ARPspoof 监控的某一台设备必须制定这台设备的 IP-MAC 地址映射，对每一台设备都要在 snort.conf 文件中用新的一行列出。映射一旦变化，就必须重配该文件。注意：在启用 ARPspoof 之前应该将 DHCP 获取 IP 的设备改成静态 IP 地址。

7. Sfportscan

Sfportscan 模块可用来应对网络攻击前的侦查（Reconnaissance）阶段的扫描行为，因为在攻击发起之初，攻击者对目标网络并不了解，为了得到目标网络的服务及协议，往往会使用传统的端口扫描工具，盲目发起扫描，以获得目标主机的回应，当然正常情况下这种回应非常少，攻击者最常用的扫描工具类似于 nmap，而 sfportscan 恰好是应对这种扫描的工具，它能应对如表 6-3 所示的扫描类型。

表 6-3　sfportscan 侦测的扫描类型

扫描类型	用途
TCP Portscan UDP Portscan IP Portscan	一台主机扫描另一台主机的多个端口
TCP Decoy Portscan UDP Decoy Portscan IP Decoy Portscan	攻击者将自己来源混合在多个虚假源地址发起的扫描, 这种策略往往可以隐藏攻击者的真实位置

（续表）

扫描类型	用途
TCP Distributed Portscan UDP Distributed Portscan IP Distributed Portscan	多台主机对一台目标主机进行扫描，往往用来欺骗 IDS 或控制主机
TCP Portsweep UDP Portsweep IP Portsweep ICMP Portsweep	一台主机扫描多台主机的某一个端口，如 80 端口

对于 sfportscan 的配置，使用 sfportscan 预处理前提是首先使用 Stream5 preprocessor，它可以指出类似于 UDP 和 ICMP 无连接协议的端口方向。

```
preprocessor sfportscan:
```

可用参数如下：

- proto <protocol>：协议可选择的值包括 TCP、UDP、IP_proto、IGMP、ALL。
- scan type <scan type>：扫描类型可选择的值包括 portscan、portsweep、decoy portscan、distributed portscan、all。
- sense level <level>：级别可选择的值包括 low、medium、high。
- watch ip < ip1|ip2/cidr[[port|port2-port3]]>：指定要被检测的 ip 及 port。
- ignore scanners < ip1|ip2/cidr[[port|port2-port3]]>：忽略来源的扫描警报。
- ignore scanned < ip1|ip2/cidr[[port|port2-port3]]>：忽略目的地扫描报警。
- logfile <file>：输出 logfile 到指定的地方，若使用相对位置，则该文件会放在 Snort 配置目录中。

下面看一下 OSSIM 系统中 snort.conf 配置文件的 396 行，如图 6-6 所示：

```
# Portscan detection.  For more information, see README.sfportscan
preprocessor sfportscan: proto  { all } memcap { 10000000 } sense_level { low }
```

图 6-6　sfportscan 配置

实际应用中通常利用 sfportscan 预处理器，找出潜在 Malware（流氓软件）。

6.1.2　调整预处理程序

Snort 实施检测是由于某些类型的通信中包含了恶意流量，消耗过多的资源，这时出现 Snort 丢包现象，这就相当于 Snort 失效一样。这种情况怎么解决呢？可以采用调整预处理程序。攻击者一次攻击中，可能展开成千上万个包，如果预处理程序由于没有分配到足够内存，而不能标准化处理它们时，这种攻击将通过 Snort，而不被检测到。

预处理程序需要消耗系统资源，比如一个预处理程序没有足够的内存来处理，一次攻击会分解成若干个包，如果分配预处理程序内存不足，就会无法展开，并处理这些包，Snort 即会遗漏攻击的具体细节。我们通过调整预处理程序便能改变这一现象。

6.1.3 网络攻击模式分类

OSSIM 中的图形化 Alarm 报警是通过数据挖掘方法，从庞大的报警事件中提取网络入侵模式，后台主要是依托的关联规则算法和序列规则算法，其核心被用户提取每个链接记录内部和记录间的特征模式。根据 Snort 扫描类型的不同将网络攻击模式分为以下几类，每一类在 OSSIM 中都有特定的图例显示。

（1）从内网发往外网的一对多模式（internal to external one-to-many）

右侧为它的图标 ATTACK PATTERN，实例如图 6-7 所示。这种情况下你可看到一个 IP 地址向多个 IP 地址发送大量数据，那么它有可能是邮件服务器也有可能是垃圾邮件僵尸，或者是网络端口扫描。

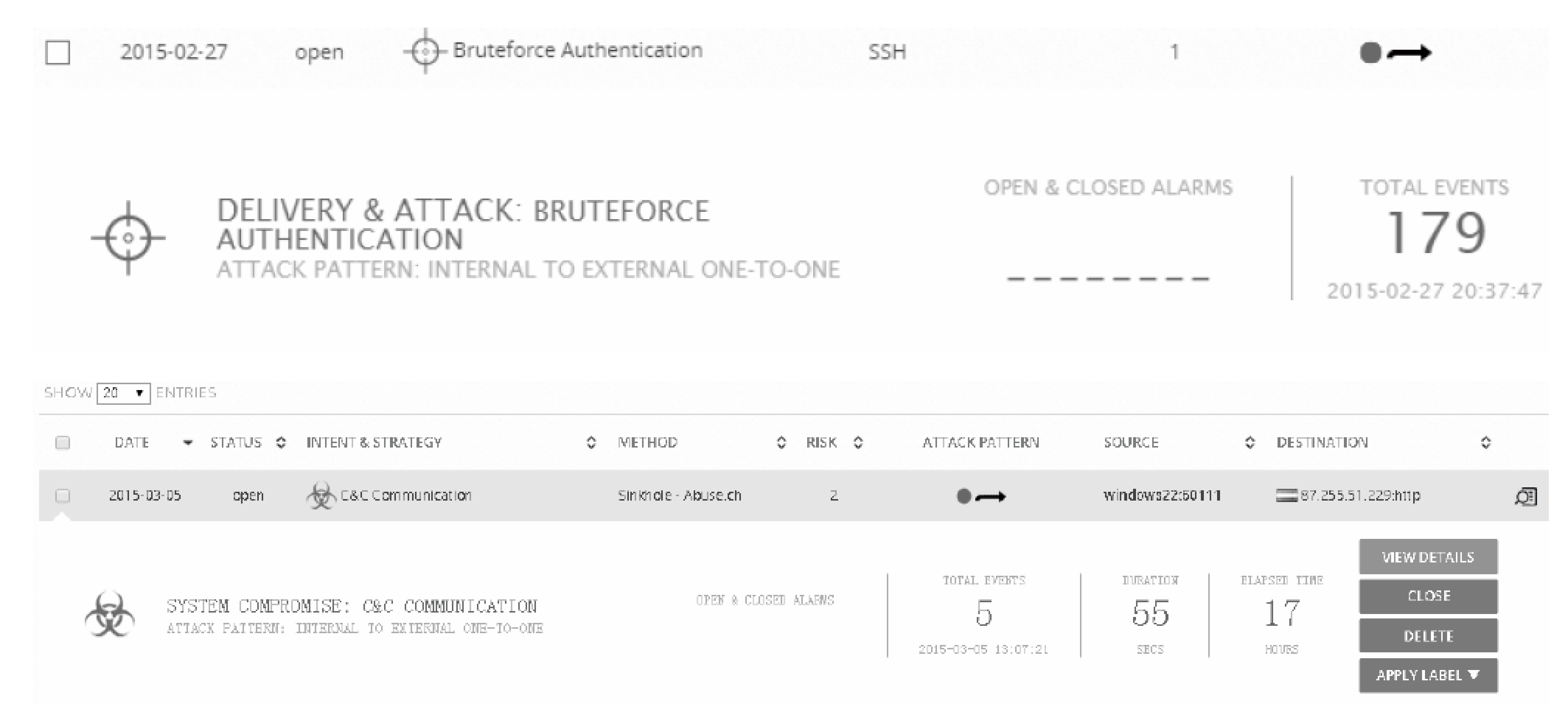


图 6-7　内网发往外网的一对多模式

（2）从外网到内网的一对一模式（external to internal one-to-one）

右侧为它的图标，实例如图 6-8 所示。这种情况下，数据从一个 IP 地址传输到另一个 IP 地址，意味着常规服务器通信，不过也有可能是针对特定系统的目标攻击。

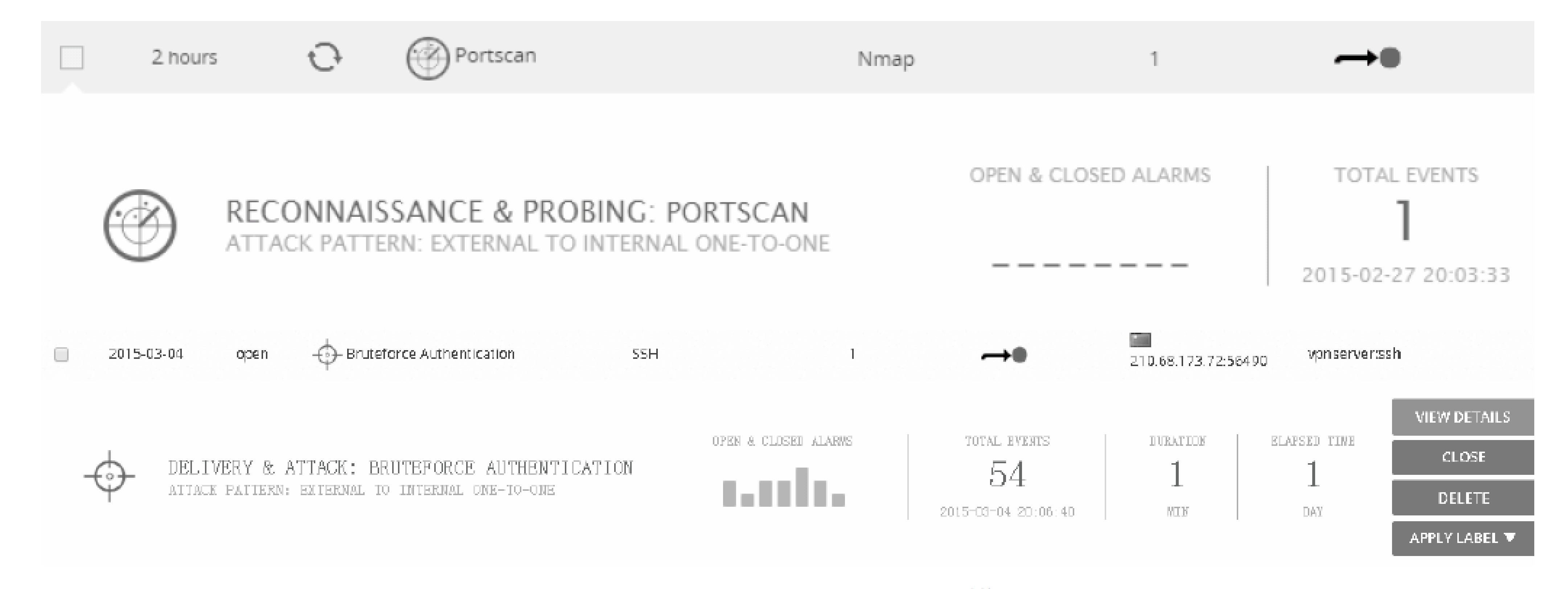


图 6-8　外网到内网的一对一模式

（3）外网间的多对一模式（external to external many-to-one）

它的图标为 ，实例如图 6-9 所示，当你发现多个 IP 地址发送数据到一个 IP 地址，这有可能是 Syslog 服务器，也有可能是针对目的 IP 地址的分布式 DOS 攻击。

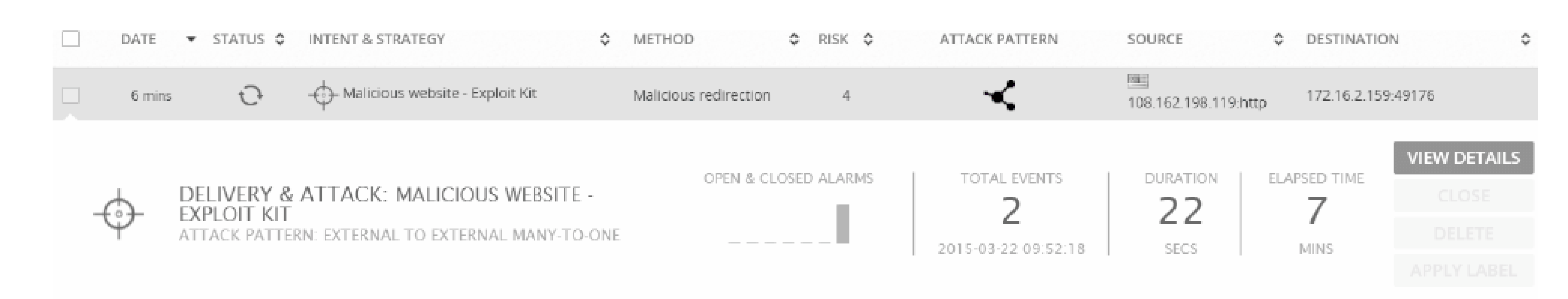


图 6-9　外网间的多对一模式

（4）内网中一对一模式（internal one-to-one）

右侧为它的图标，实例如图 6-10 所示。

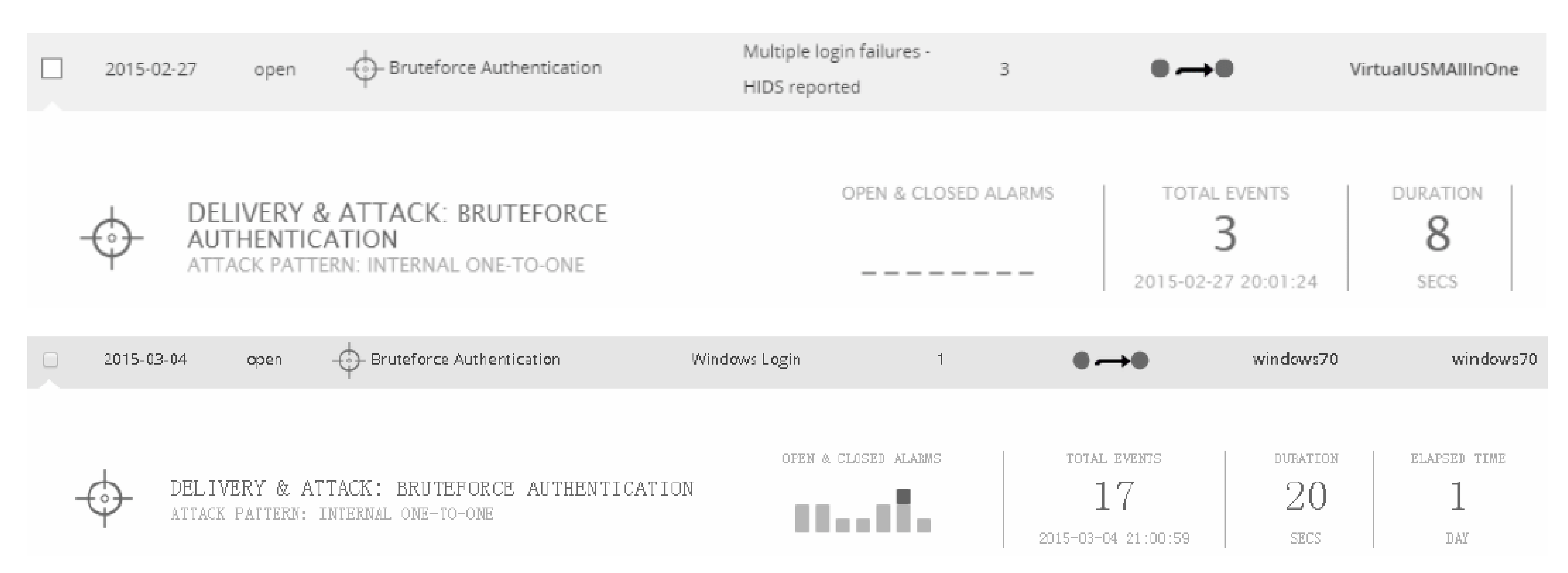


图 6-10　内网中一对一模式

（5）外部到外部的一对一模式（external to external one-to-one）

右侧为它的图标 ，实例如图 6-11 所示。

图 6-11　外部到外部的一对一模式

通过以上 5 种模式的图形化显示，今后管理员无须仔细分析日志，只要观察图像就能快速分辨出攻击类型。

6.2 Snort 日志分析利器

要深入分析 OSSIM 中 SIEM 控制台显示的内容，首先要知道 Snort 的日志分析方法，我们先看看下面 3 种工具：

- ACID：老款管理员控制台。
- BASE：被称为基本的分析和安全引擎，它是基于 PHP 的分析引擎，可以搜索、处理由 IDS、防火墙、网络监视工具所生成的安全事件数据。由于 BASE 目前已停止升级，采用 Suricata+Barnyard2+BASE 是当下较为流行的一种轻量级 IDS 系统，这种系统的搭建是在 Snort+ACID 架构基础上演变而来的，需要的软件包有 Suricata、Barnyard2、Base、Yaml、Adodb、Snort 的规则库等。
- Sguil：这是一款被称为网络安全专家监视网络活动的控制台工具，它可以用于网络安全分析。其主要部件是一个直观的 GUI 界面，可以为 Snort/Barnyard 提供实时的事件活动监控。用于网络安全监视（Network Security Monitoring ，简称 NSM）活动和 IDS 警告的事件驱动分析。

这几种工具安装有一定难度 即便是熟悉这种架构和安装方法的工程师也需要费一番功夫进行安装调试。如果大家不愿手工安装这些工具，那么可以使用 Security Onion，它集成了以上三款工具，基于 ISO 镜像的安装方式方便部署，为初学者节省了不少时间，有关它的使用方法大家可以到作者博客继续学习。

6.3 Snort 日志分析

当 Snort 启动后，会不停地抓取网络上的数据包，因此它会在硬盘上记录大量的报警信息。未处理的大量日志信息对用户来讲是无意义的，因此，需要对日志文件的内容进行分析，从无序日志中获取有价值的信息，这样才可以帮助用户针对攻击威胁采取必要措施。

Snort 的日志一般位于/var/log/snort/目录下。可以通过修改配置文件来设置 Snort 的报警形式。基于文本的格式、Libpcap 格式和数据库存储是 Snort 最重要的 3 种报警形式。本节主要对各种报警形式及其配置进行说明。

6.3.1 工作模式

Snort 具有 3 种工作模式，分别为嗅探器模式、分组日志模式与网络入侵检测模式。

（1）嗅探器模式

Snort 使用 Libpcap 包捕获库，在这种模式下，Snort 使用网络接口的混杂模式读取并解析

数据包。该模式使用命令如下：

```
#snort -v
```

命令执行结果如下：

```
alienvault:~# snort -v |more
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.0.4 (Build 111)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t
eam
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 7.6 2008-01-28
           Using ZLIB version: 1.2.3.3

Commencing packet processing (pid=6569)
07/09-14:22:17.240957 10.32.14.133:22 -> 10.32.14.131:64648
TCP TTL:64 TOS:0x10 ID:18173 IpLen:20 DgmLen:600 DF
***AP*** Seq: 0x76EE71AE  Ack: 0x3AA37122  Win: 0x11  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

===============================================================================
Run time for packet processing was 2.336315 seconds
Snort processed 97 packets.
Snort ran for 0 days 0 hours 0 minutes 2 seconds
   Pkts/sec:           48
===============================================================================
Packet I/O Totals:
   Received:           97
   Analyzed:           97 (100.000%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            0 (  0.000%)
   Injected:            0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:           97 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:           95 ( 97.938%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:           22 ( 22.680%)
        TCP:           73 ( 75.258%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
```

注意：这里的参数是小写字母 v，而大写 V 则是显示 snort 版本。以上结果只显示 TCP/IP 网络数据包头信息，如果想查看详细的应用层数据信息，则需要输入以下命令：

```
#snort -vd
```

命令执行结果如下：

```
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.0.4 (Build 111)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 7.6 2008-01-28
           Using ZLIB version: 1.2.3.3

Commencing packet processing (pid=8133)
07/09-14:31:45.391925 10.32.14.133:22 -> 10.32.14.131:64648
TCP TTL:64 TOS:0x10 ID:18488 IpLen:20 DgmLen:600 DF
***AP*** Seq: 0x76F06DFE  Ack: 0x3AA38F72  Win: 0x13  TcpLen: 20
7E 98 DA 15 0A 27 FD B5 64 F2 90 71 25 61 80 3F  ~....'..d..q%a.?
09 3B 01 5B B4 41 7C 63 BF CF DD 2E 4D 35 BA 16  .;.[.A|c....M5..
68 41 F5 31 1D 51 2A F7 04 8E F0 42 D2 8E F5 78  hA.1.Q*....B...x
11 26 91 FC 00 54 E7 10 71 6F C8 C4 18 1D 84 C6  .&...T..qo......
04 73 F2 1F 9F EB AF 53 A5 58 ED F2 5B 14 46 1A  .s.....S.X..[.F.
```

如果要查看数据链路层的包头信息，输入“snort -vde”：

```
#snort  -vde
```

命令执行结果如下：

```
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.0.4 (Build 111)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 7.6 2008-01-28
           Using ZLIB version: 1.2.3.3

Commencing packet processing (pid=10471)
07/09-14:45:38.582467 08:00:27:1F:37:49 -> 34:17:EB:99:4C:77 type:0x800 len:0x266
10.32.14.133:22 -> 10.32.14.131:64648 TCP TTL:64 TOS:0x10 ID:19247 IpLen:20 DgmLen:600
***AP*** Seq: 0x76F6006E  Ack: 0x3AA3B5D2  Win: 0x15  TcpLen: 20
D3 B6 00 47 A4 EA 98 5A 71 46 93 BA 78 14 1F A8  ...G...ZqF..x...
DB 24 41 F1 74 A5 DA 8C 4A 63 DE 32 7F BE A5 F8  .$A.t...Jc.2....
9E 54 B9 A7 0D 73 F0 F1 DA CC 5D FD A3 19 A3 47  .T...s....]....G
```

使用MAC地址表示

还有一个类似“-d”参数，“-X”会从数据链路层开始输出原始数据包。

如果将 Snort 作为 IDS 使用，不建议在命令行下使用“-vd”，尤其是“-ved”参数，因为详细模式将数据包信息打印到控制台，这样严重影响 Snort 的性能，并很容易引起丢包。

（2）数据包记录模式

如果想把数据信息记录到磁盘上的某个文件，那就需要使用 Packet logger（包记录）模式。命令如下：

```
#snort -ved -l  ./log
```

这时 Snort 会把数据链路层、TCP/IP 报头及应用层信息写入当前目录 log（log 目录已建立）下的 snort.log.140493321 文件中，而且文件格式为二进制文件。你也许会问，ASCII 格式的日志文件格式非常好识别，为什么不直接记录成 ASCII 格式呢？因为系统本身记录的格式就是二进制的，如果再转换成我们能识别的 ASCII 格式无疑会加重系统负荷，所以 Snort 在做 IDS

使用时，应采用二进制格式记录，另外还要注意：“-l”参数是小写字母 l。

如果想查看所记录的日志，就得使用“r”参数，操作实例如下：

```
#snort -dvr snort.log.140493321
```

还可以提取部分感兴趣的数据，例如只读取 ICMP 包，输入如下命令：

```
#snort -dvr snort.log.140493321 icmp
```

只读取 tcp 包，输入如下命令：

```
#snort -dvr snort.log.140493321 tcp
```

如果想记录某个网段的数据呢，操作命令如下：

```
# snort -vde -l ./log -h 10.32.14.0/24
```

下面进行比较复杂的实验，环境为 OSSIM 3.1，操作演示如图 6-12 所示。

首先在控制台上启动 snort：

```
#snort -l /var/log/snort/  -c /etc/snort/snort.eth0.conf
```

当需要结束命令时按下 Ctrl+C 组合键，收到警报如图 6-13 所示。

```
| Patterns           : 13406
| Match States       : 10678
| Memory (MB)        : 45.05
|   Patterns         : 0.99
|   Match Lists      : 2.34
|   DFA
|     1 byte states : 1.37
|     2 byte states : 40.13
|     4 byte states : 0.00
+----------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 3506 ]
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Reload thread starting...
Reload thread started, thread 0xb458bb90 (19018)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.0.4 (Build 111)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 7.6 2008-01-28
           Using ZLIB version: 1.2.3.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.13  <Build 18>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=19018)
^X
```

图 6-12　记录某网段数据

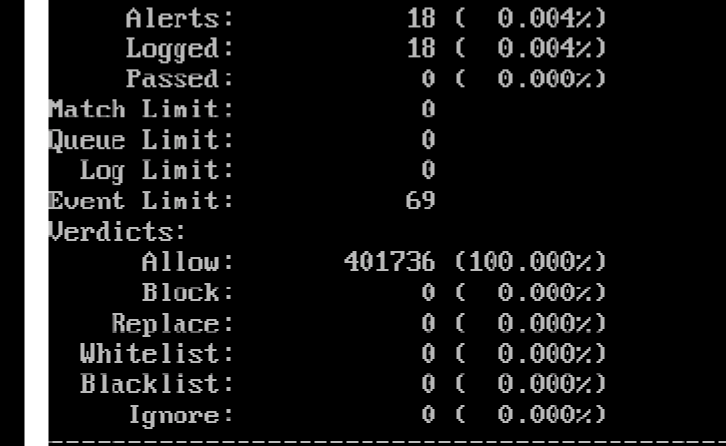


图 6-13　收到 Alerts

Alerts 数量就是在日志里看到的记录数量，这两者一致。

在另一个控制台查看日志情况。

```
#tail -f /var/log/auth.log
```

```
Jul 14 22:58:22 alienvault snort: [1:2008428:9] ET TROJAN Suspicious User-Agent (HTTP Downloader) [Classification: A Network Trojan was Dete
3719 -> 112.90.17.159:8080
Jul 14 22:58:22 alienvault snort: [1:2008428:9] ET TROJAN Suspicious User-Agent (HTTP Downloader) [Classification: A Network Trojan was Dete
3720 -> 112.90.17.159:8080
Jul 14 22:58:30 alienvault snort: [1:2009967:5] ET P2P eMule KAD Network Connection Request [Classification: Potential Corporate Privacy Vio
:20508 -> 88.137.82.81:32047
Jul 14 22:58:38 alienvault snort: [1:2014703:7] ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port Reserved Bit Set - Likely
Kazy [Classification: Potential Corporate Privacy Violation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 194.2.0.20:53
Jul 14 22:58:40 alienvault snort: [1:2009968:4] ET P2P eMule KAD Network Connection Request(2) [Classification: Potential Corporat
e Privacy Violation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 90.175.46.181:50504
Jul 14 22:59:01 alienvault CRON[17977]: pam_unix(cron:session): session opened for user root by (uid=0)
Jul 14 22:59:01 alienvault CRON[17979]: pam_unix(cron:session): session opened for user root by (uid=0)
Jul 14 22:59:02 alienvault CRON[17979]: pam_unix(cron:session): session closed for user root
Jul 14 22:59:02 alienvault CRON[17977]: pam_unix(cron:session): session closed for user root
Jul 14 22:59:12 alienvault snort: [1:2009968:4] ET P2P eMule KAD Network Connection Request(2) [Classification: Potential Corporat
e Privacy Violation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 173.73.28.174:7128
Jul 14 22:59:12 alienvault snort: [1:2009968:4] ET P2P eMule KAD Network Connection Request(2) [Classification: Potential Corporat
e Privacy Violation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 1.34.33.40:5672
Jul 14 22:59:30 alienvault snort: [1:2009970:4] ET P2P eMule Kademlia Hello Request [Classification: Potential Corporate Privacy V
iolation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 86.202.198.33:35624
Jul 14 22:59:30 alienvault snort: [1:2009970:4] ET P2P eMule Kademlia Hello Request [Classification: Potential Corporate Privacy V
iolation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 60.16.12.103:14670
Jul 14 22:59:30 alienvault snort: [1:2009970:4] ET P2P eMule Kademlia Hello Request [Classification: Potential Corporate Privacy V
iolation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 125.87.45.224:18354
Jul 14 22:59:30 alienvault snort: [1:2009970:4] ET P2P eMule Kademlia Hello Request [Classification: Potential Corporate Privacy V
iolation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 79.2.52.82:4672
Jul 14 22:59:30 alienvault snort: [1:2009970:4] ET P2P eMule Kademlia Hello Request [Classification: Potential Corporate Privacy V
iolation] [Priority: 1] {UDP} 10.32.14.131:20508 -> 124.64.58.147:2650
```

此时，我们可以在 SIEM 控制台中看到如图 6-14 所示。

Security Events (SIEM)　　Manage References

Pause　　Paused.

Date	Event Name	Risk	Generator	Sensor	Source IP	Dest IP
2014-07-14 23:04:34	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 23:03:35	snort: "ET P2P eMule Kademlia Hello Request"	0	snort	alienvault	Host-10-32-14-131:20508	27.155.180.32:23362
2014-07-14 23:03:35	snort: "ET P2P eMule Kademlia Hello Request"	0	snort	alienvault	Host-10-32-14-131:20508	119.119.178.251:14743
2014-07-14 23:03:35	snort: "ET P2P eMule Kademlia Hello Request"	0	snort	alienvault	Host-10-32-14-131:20508	186.134.25.210:13767
2014-07-14 23:03:35	snort: "ET P2P eMule Kademlia Hello Request"	0	snort	alienvault	Host-10-32-14-131:20508	81.56.74.75:31530
2014-07-14 23:03:35	snort: "ET P2P eMule Kademlia Hello Request"	0	snort	alienvault	Host-10-32-14-131:20508	82.56.186.140:4672
2014-07-14 22:59:30	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:30	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:30	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:30	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:30	ossec: Multiple IDS events from same source ip.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:14	snort: "ET P2P eMule KAD Network Connection Request(2)"	0	snort	alienvault	Host-10-32-14-131:20508	173.73.28.174:7128
2014-07-14 22:59:14	snort: "ET P2P eMule KAD Network Connection Request(2)"	0	snort	alienvault	Host-10-32-14-131:20508	1.34.33.40:5672
2014-07-14 22:59:12	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault
2014-07-14 22:59:12	ossec: IDS event.	0	ossec-ids	alienvault	Host-10-32-14-131	alienvault-alienvault

Date: 2014-07-14 22:59:30
Event: ossec: IDS event.
Risk: 0
Plugin: ossec-ids, Plugin_sid: 20101
Sensor: alienvault [10.32.14.133]
Source IP: Host-10-32-14-131 [10.32.14.131]
Dest IP: alienvault-alienvault [10.32.14.133]
Priority: 1, Reliability: 1
Interface: eth0, Protocol: tcp
Asset Src: 2, Asset Dst: 2

Filters

图 6-14　SIEM 控制台

（3）HIDS 模式

HIDS 是大家需要掌握的重点，这种模式集成了嗅探模式和日志模式，并且需要载入规则库才能正常工作。操作方式如下：

```
#snort -vde -l ./log -h 10.32.14.0/24 -c /etc/snort/snort.eth0.conf
```

当前目录下有 log 目录，如果指定了“-l ./log”参数，系统会默认将日志存在/var/log/snort/目录下。snort.conf 文件中还包含了指定检测规则的具体路径。

6.3.2　输出插件

输出插件的作用是将报警输出到屏幕或转储到文件。所以对 Snort 而言，输出插件为系统的瓶颈，Snort 本身能对封包进行快速读取和分析处理，但是试图将其显示输出或者存储到数据库中时却有些力不从心。如何将 Snort 日志记录到一个指定文件呢？我们通过执行以下命令（假设/var/log/snort/yourfile 文件存在）来记录。

```
#snort -L yourfile
```

那么，此时系统自动将 snort 日志保存到/var/log/snort/yourfile 文件中。后续文章我会给大家一个方法,配置 Snort 统一格式输出并且有 Snort 的标准日志应用 Barnyard 负责输出的方法,这里我们先看看 Snort 的输出插件。

1. -A alert-mode

在 Snort 入侵检测模式中 alert-mode 包括 fast、full、unsock 和 none 4 种模式。

（1）fast

fast 是一种快速简单的输出插件,之所以快是因为它只记录 Timestamp(时间戳)、Signature（特征）、Source IP、DestinationIP、Source port、Destination port、TCP flags 和 Protocol。使用方法如下：

```
#snort -A fast
```

```
alienvault:~/log/log# snort -A fast |more
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.0.4 (Build 111)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 7.6 2008-01-28
           Using ZLIB version: 1.2.3.3

Commencing packet processing (pid=4444)
07/09-17:20:31.452368 10.32.14.133:22 -> 10.32.14.131:64648
TCP TTL:64 TOS:0x10 ID:44794 IpLen:20 DgmLen:600 DF
***AP*** Seq: 0x7993ABBE  Ack: 0x3AAEA4B2  Win: 0x2C  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

上面的命令不保存到文件，如果希望保存到文件（当不使用-l 参数，默认保存位置是/var/log/snort/），命令如下：

```
#snort -A fast -h 10.32.14.0/24 -c /etc/snort/snort.eth0.conf
```

（2）full

full 将每个产生警报的 IP 解码后的包记录下来。与 fast 不同的是，full 记录的更全面。full 为预设告警模式。

（3）unsock

这个插件的作用是建立一个 UNIX 域管道并向它发送警报。当然其他进程也可对该管道进行监听，目的是实时接收 Snort 警报数据。

```
#snort -A unsock
```

（4）none

这个插件作用是关闭警报。大家只需掌握“-A”后面跟的 4 个参数的含义，了解这 4 个参数只是让 Snort 能以不同的方式报警。

2. tcpdump 格式输出

有时候需要将 Snort 日志输出成 tcpdump 文件格式的包，因为这样可以让多种应用程序和工具读取 tcpdump，这时我们如下操作（以 OSSIM 3.1 系统为例）：

（1）编辑 snort.conf.

```
#vi /etc/snort/snort.eth0.conf
```

找到 390 行，启用 output log_tcpdump: tcpdump.log。配置参数如图 6-15 所示。

```
374 # unified2
375 # Recommended for most installs
376 # output unified2: filename snort_eth0, limit 128, nostamp, mpls_event_types, vlan_event_types
377 output unified2: filename snort_eth0, limit 128
378
379 # Additional configuration for specific types of installs
380 # output unified2: filename snort_eth0, limit 128, nostamp
381 # output unified2: filename snort_eth0, limit 128, nostamp
382
383 # syslog
384 output alert_syslog: LOG_AUTH LOG_ALERT
385
386 # logging to a csv file
387 output alert_csv: csv.out timestamp,msg,srcip,sport,dstop,dport,protoname,itype,icode
388
389 # pcap
390 output log_tcpdump: tcpdump.log
```

图 6-15　输出格式参数

（2）保存退出，并重启 Snort 服务使其配置生效。

（3）查看 tcpdump.log。

```
alienvault:/var/log/snort# ls -l
-rw------- 1 root  adm    504 Jul 14 22:02 tcpdump.log.1405389707
-rw------- 1 root  adm   3410 Jul 14 22:45 tcpdump.log.1405392066
```

（4）CSV 格式输出。

采用 CSV 格式的目的主要考虑向其他数据库或电子表格软件输入。CSV（Comma Separated Value）是用逗号分隔值的，为文本文件。CSV 格式可以记录 24 个字段，如表 6-4 所示。

表 6-4　规则选项关键字含义

序号	关键字	作用
1	timestamp	时间戳
2	msg	特征码名称
3	proto	协议
4	src	源地址
5	srcport	源端口
6	dst	目标地址
7	dstport	目标端口
8	ethsrc	源 MAC
9	ethdst	目标 MAC
10	ethlen	以太网帧长度
11	tcpflags	TCP 标志位
12	tcpseq	TCP 序列号
13	tcpack	TCP ack 号
14	tcplen	TCP 长度
15	tcpwindow	tcp 窗口值
16	ttl	ip 头的 ttl 的值
17	tos	ip 头中 TOS 字段的值
18	id	ip 头的分片 id 值
19	dgmlen	数据报的总长度，包括数据报头和数据报文
20	iplen	IP 包长度
21	icmptype	ICMP 类型
22	icmpcode	ICMP 代号，默认为 0
23	icmpid	ICMP 报文 IP 头的 ID，默认是随机的
24	icmpseq	ICMP ECHO 顺序号的值

实现 csv 格式输出时，首先编辑 snort_eth0.conf 文件。在其中添加:output alert_csv: csv.out，后面默认有 24 个关键字，我们也可按上图的参数输入。

查看 csv 输出：

```
alienvault:/var/log/snort# tail -f csv.out
07/15-00:29:51.254425 ,"GPL SHELLCODE x86 inc ebx NOOP",,,,,,,
```

```
    07/15-00:29:51.357178 ,"GPL SHELLCODE x86 inc ebx NOOP",,,,,,,
    07/15-00:29:54.541833 ,"ET P2P eMule Kademlia Hello Request",,,,,,,
    07/15-00:29:54.541846 ,"ET P2P eMule Kademlia Hello Request",,,,,,,
    07/15-00:31:55.606599 ,"ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port
Opcode 6 or 7 set - Likely Kazy",,,,,,,
```

以下为 24 个参数全部启用的效果。默认为全部启用，除非指定具体参数。

在 snort_eth0.conf 加入下面一行内容。

```
    output alert_csv: csv.out
```

表示将表 6-4 中所列 24 个参数都输出到 csv.out 文件中，下面我们查看/var/log/snorty/csv.out 的内容：

```
alienvault:/var/log/snort# tail -f /var/log/snort/csv.out
07/15-02:33:34.813791 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,112.86.42.91,38655,34:17:EB:99:4C:
77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,2506,66,67584,,,,
07/15-02:33:34.814402 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,113.233.174.149,57285,34:17:EB:99:
4C:77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,2507,66,67584,,,,
07/15-02:33:34.813472 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,115.239.80.6,27402,34:17:EB:99:4C:
77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,2505,66,67584,,,,
07/15-02:33:34.813795 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,112.86.42.91,38655,34:17:EB:99:4C:
77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,2506,66,67584,,,,
07/15-02:35:36.487186 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,118.232.160.24,17192,34:17:EB:99:4
C:77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,5346,66,67584,,,,
07/15-02:35:36.487335 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,183.44.145.219,19364,34:17:EB:99:4
C:77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,5347,66,67584,,,,
07/15-02:39:39.832498 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,1.28.188.143,25379,34:17:EB:99:4C:
77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,11026,66,67584,,,,
07/15-02:39:39.832506 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,60.255.0.20,27915,34:17:EB:99:4C:7
7,58:8D:09:52:DB:FF,0x50,,,,,,64,0,11027,66,67584,,,,
07/15-02:41:41.507703 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,120.0.74.172,24144,34:17:EB:99:4C:
77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,13672,66,67584,,,,
07/15-02:42:42.342802 ,1,2009970,4,"ET P2P eMule Kademlia Hello Request",UDP,10.32.14.131,20508,79.18.120.234,38715,34:17:EB:99:4C
:77,58:8D:09:52:DB:FF,0x50,,,,,,64,0,15064,66,67584,,,,
```

查看 SIEM 控制台输出。当设置成功 CSV 输出后，可以查看 OSSIM 3 系统下 SIEM 控制台的日志输出，如图 6-16 所示。

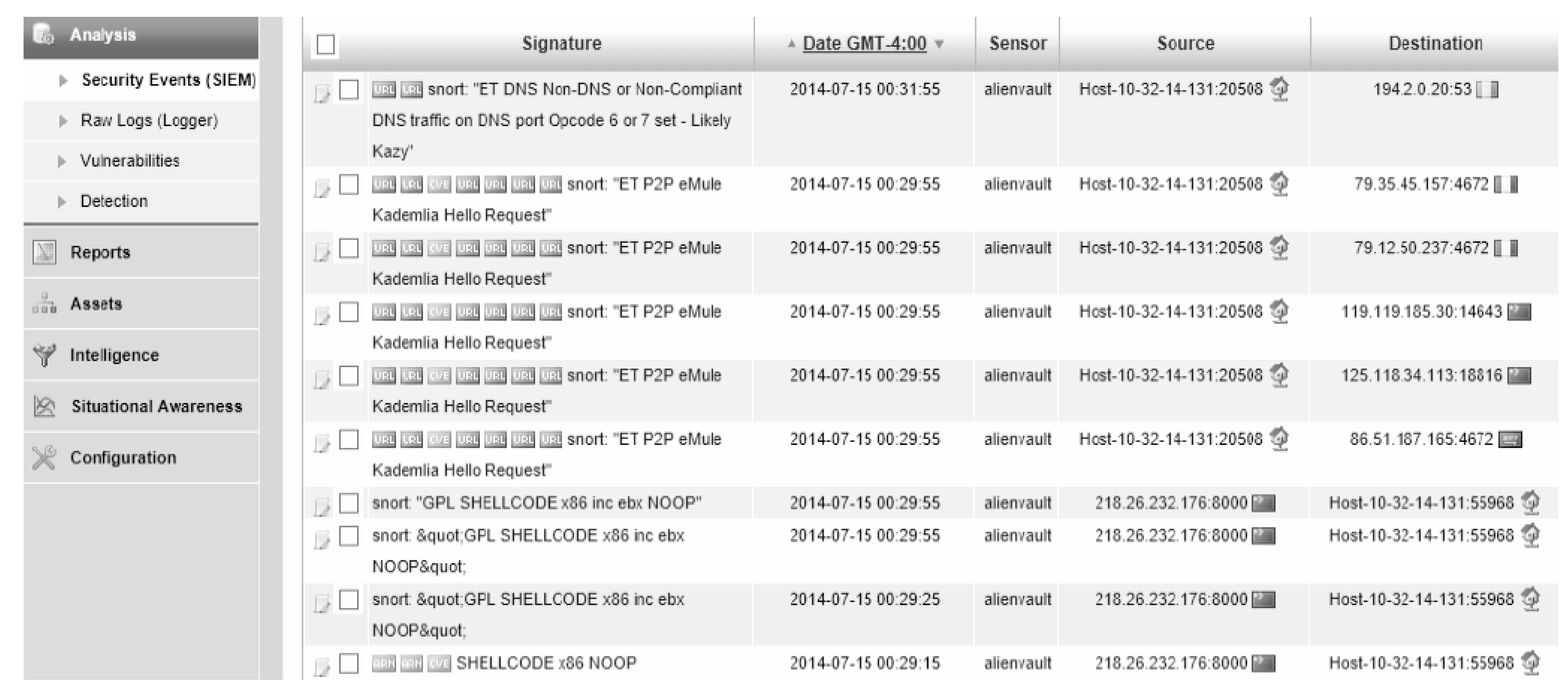


图 6-16　SIEM 控制台日志输出

经过以上实验发现两种不同警报方式的结果都相同，很显然 OSSIM 3 系统的 SIEM 控制台显示出的警报更人性化。

（5）Syslog 输出

Syslog 服务器能从各种服务和设备（网络设备，Web 服务器等）收集日志信息。将 Snort 警报数据写入 Syslog 服务器对于分析网络入侵事件很有帮助。所以这也是 Snort 一个比较重要的插件之一。最简单方法是打开 snort_eth0.conf 将 384 行的 output alert_syslog: LOG_AUTH LOG_ALERT 这行启用，并重启 Snort 即可生效。我们使用如下命令观察 Snort 发出的警报：

```
#tail -f /var/log/auth.log
```

Syslog 有三个配置选项：

- facility
- priority
- options

启用选项的命令是在上面命令的基础上加上这 3 个参数。

```
output alert_syslog: LOG_AUTH LOG_ALERT facility  priority options
```

保存退出并重启 Snort 即可。

这时，如果启用 csv 输出模式，可以将 Syslog 和 csv 输出做个对比，如图 6-17 所示，图中上部分为 csv 输出，下部分为 auth.log 输出。

图 6-17　输出结果比较

我们发现，同样一个攻击事件的报警，在它们的输出中，除了格式不统一以外，其他都相同。

（6）数据库输出

利用数据库输出插件可以将日志保存到数据库中，它支持 MySQL。当数据库插件被关联到数据库后，就可以实现对 Snort 报警进行分类、查询和按优先级组织排序，甚至利用应用程序对数据库中的告警数据通过 GUI 界面展现给用户。

在千兆网络环境下需要考虑性能问题，数据库插件有可能影响到 Snort 性能，插件自身并不是瓶颈，当大量数据写入数据库时，必须等待磁盘 I/O 响应，这时就会受到影响。如果通过网络将日志转发到另一台主机的数据库中，那么延迟会比较大。后面我们会采用 Barnyard+Unified 的方式解决这种性能问题。

解决这种性能问题的主要思路是让 Snort 采用 Unified 的格式存储，以它的最大速度处理输出数据，而不是像传统方式要等到写盘完成后，再继续操作，这样浪费了大量的时间。Barnyard 能将二进制数解析成与它能够识别的格式，并且是完全独立于 Snort 运行。警报被立刻写入数据库，并且不影响 Snort 的抓包能力。所以说这种组合适合于大流量环境，OSSIM 环境配置如图 6-18 所示。我们还是查看/etc/snort/snort.eth0.conf 配置文件，大约 392 行的位置。

```
392 # database
393 # output database: alert, <db_type>, user=<username> password=<password> test dbname=<name> host
    =<hostname>
394 # output database: log, <db_type>, user=<username> password=<password> test dbname=<name> host=<
    hostname>
```

图 6-18　数据库输出配置

此处有两个重要的选项，分别是 alert 和 log。393 行可以选择将警报数据写入数据库，394 行可以同时将日志数据也写入数据库。下面我们看看如何使用数据库插件，对照上面参数配置如下选项：

- <db_type>为 mysql。
- User=<username>为 snortdb_username 或者为 Snort 传感器创建的 MySQL 用户名。
- Password=<password> 为 snortdb_password 或为 Snort 传感器创建 snortdb_username 的 MySQL 口令。
- Dbname=<name>为 snortdb 或者自己指定的入侵数据库名称。
- Host=<hostname>为本机环路地址 127.0.0.1。

（7）输出 unixsock

通过 Alert_unixsock 打开 UNIX 套接字，并且把报警信息发送到那里。外部的程序 / 进程会在这个套接字上侦听并实时接收这些报警数据。

例子：

```
output alert_unixsock
```

加入到 snort.eth0.conf 配置文件后，重启 Snort 服务即生效。也可以通过以下命令手动启动：

```
# snort -c /etc/snort/snort.eth0.conf
```

此时会在/var/log/snort/下产生 snort_alert 文件，配置如图 6-19 所示。

```
-rw------- 1 root  adm    5536 Jul 14 23:20 snort.1405393451
-rw------- 1 root  adm       0 Jul 14 23:57 snort.1405396621
-rw------- 1 root  adm       0 Jul 15 00:08 snort.1405397291
-rw------- 1 root  adm     950 Jul  9 14:08 snort.log.1404929215
-rw------- 1 root  adm     587 Jul  9 19:36 snort.log.1404948922
srwxr-xr-x 1 root  adm       0 Jul 15 03:37 snort_alert
-rw-r----- 1 snort adm 3388734 Jul 10 06:22 snort_eth0.1404901604
-rw-r----- 1 snort adm   27164 Jul 10 11:36 snort_eth0.1404987971
```

图 6-19　snort_alert

为了演示这一功能，我们通过一小段程序（功能是本地进程间的通信）来看看效果如何。如图 6-20 所示，是一段 Perl 脚本，其主要功能是将 Snort 传送的数据报文的报警名称通过读取 unixsock 信息实现打印到屏幕。

```
#!/usr/bin/perl
# Include the socket libraries
use IO::Socket;
# This is the template to capture the Alert Name
# Edit this to get the additional packets.
$TEMPLATE = "A256 A*";
# Release the socket if it already exists
unlink "/var/log/snort/snort_alert";
# In case of user termination - exit gracefully.
$SIG{TERM} = $SIG{INT} = sub { exit 0 };
# Open up the socket.
my $client = IO::Socket::UNIX->new(Type => SOCK_DGRAM,
                  Local => "/var/log/snort/snort_alert")
  or die "Socket: $@";
print STDOUT "Socket Open ... \n";
# Loop receiving data from the socket, pulling out the
# alert name and printing it.
my $data;
while ( true ) {
    recv($client,$data,1024,0);
    @FIELDS = unpack($TEMPLATE, $data);
    print "@FIELDS[0] \n";
}
# At termination close up the socket again.
END {unlink "/var/log/snort/snort_alert";}
```

图 6-20　脚本

执行该 Perl 程序，结果如图 6-21 所示。我们看到显示在屏幕上出现了 5 条警报。

```
alienvault:/tmp# ./test.pl
Socket Open ...
ET P2P eMule Kademlia Hello Request
ET P2P eMule Kademlia Hello Request
ET P2P eMule Kademlia Hello Request
ET P2P eMule Kademlia Hello Request
ET P2P eMule Kademlia Hello Request
```

图 6-21　运行演示

（8）Unified 格式输出

前面介绍了多种输出插件，它们各有利弊，下面介绍 OSSIM 系统中常用的 unified 插件。这种插件的最大特点是速度快，它能快速输出 Snort 报警信息和日志信息。它输出两类文件：警报文件和数据包日志文件，警报文件仅记录摘要信息（内容包括源 IP、目的 IP、协议、源端口、目的端口、报警消息 ID），日志文件包含完整的包信息，用户可以自行下载。

设置方法是启用以下配置语句：

```
Output unified2: filename snort_eth0, limit 128
```

Filename 后设置输出警报文件名称为 snort_eth0。

Limit 代表输出文件允许的最大长度，默认值为 128MB。

6.4 Snort 规则编写

Snort 规则特征库中的每条规则都对应一条网络攻击的特征，Snort 通过攻击特征来识别

网络发生的攻击行为，大家在能够看懂 Snort 规则的基础上才能进行修改。本节我们讲解分析规则的基础知识。

6.4.1 Snort 规则分析

Snort 规则库中的每条规则都对应于一种入侵方式，每条规则由规则头和规则体选项组成。规则头包含有匹配的行为动作、协议类型、源 IP 及端口、数据包方向、目标 IP 及端口；规则选项则包含报警信息、内容选项等，规则选项是规则匹配的核心部分。如下面所示的一条简单的规则，括号外面的部分是规则头部，括号里面的部分是规则选项。注意：本节所有 Snort 规则在 OSSIM 系统中均可找到代码。

```
    alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP"; dsize:0;
itype:8; reference:arachnids,162; classtype:attempted-recon; sid:469; rev:3;)
```

刚看到这段代码感觉很复杂，下面解释一下这段代码的含义：如果检测到 ICMP 包，不论是来自$EXTERNAL_NET 被定义为（default = any）或$HOME_NET 被定义为（default=any），只要数据大小（dsize）是 0 并且 ICMP 类型 itype 是 8（表示回应 request），就发出告警。除了告警还包括审查功能。在 snort.conf 中有两个是用来指定监听网络的参数：

（1）var HOME_NET any，用来指定网络监听界面，默认的 any 代表任何一个本机网络界面都监听。

（2）var EXTERNAL_NET any，用来设定对外网络监听界面，默认 any 代表任何一个本机网络界面都监听。

Snort 规则是文本格式，它存在于/etc/snort/rules 目录中，一旦触发规则，Snort 会有 5 种动作类型：

- pass 动作：pass 将忽略当前的包，后续被捕获的包将被继续分析。
- alert 动作：alert 将按照已配置的格式记录包进行报警。这是常用的动作。
- dynamic 动作：它保持为一种潜伏状态，直到 activate 类型的规则将其触发，之后它将像 log 动作一样记录数据包。
- log 动作：将按照已配置的格式记录数据包;
- activate 动作：当被规则触发时产生报警，并启动相关 dynamic 类型规则，尤其在检测复杂入侵攻击时常用。

如上面那条规则例子中，每条规则包含多个规则选项，如 msg、flow、content、reference 等。各个规则选项之间用分号（；）分隔开来，而规则选项名和规则选项内容之间由冒号（：）区分。在庞大的规则库中，各个规则之间是“或”的关系，只要一个数据包与单条规则匹配，就判定该数据包为有害数据包；而每条规则中的各个规则选项是“与”的关系。只有当一个数据包与一条规则中的所有规则选项都匹配时，才能判定该数据包为有害数据包，只要有一条不符合，就不与该规则匹配。

目前 Snort 支持 ICMP、TCP、IP 和 UDP 4 种协议。规则头的后面部分则用于指定该规则的源和目的 IP 地址。Snort 规则体部分由若干个被分号隔开的片段组成，每个片段均定义了一个选项和相应的选项值。规则体中的可选项为:

- content：寻找一个指定模式下的包负载；
- flags：测试指定的 TCP 标识；
- ttl：检查 IP 头的 TTL 域的值；
- itype：ICMP 类型域匹配；
- icode：ICMP 代码域匹配；
- minfrag：设定 IP 分片大小的起始值；
- id：测试指定的 IP 头的值；
- ack：查找一个已经确认的 TCP 头；
- seq：记录一个 TCP 头顺序的值；
- logto：将匹配的包记录到指定文件的名字；
- dsize：匹配包负载的尺寸；
- offset：用于内容选项的修改，在数据包负载中设置偏移量以开始内容搜索；
- depth：用于内容选项的修改，设置搜索开始位置的字节数字；
- msg：设置当一个包产生事件时要发送的信息；
- content-list：在报文中搜索与 Snort 规则库中相匹配模式的集合；
- uricontent：对特定的 Web 协议进行通信分析，只匹配 URL 请求的部分；
- nocase：说明该规则对文本内容与规则库进行匹配时，字符串的大小写也可以混用。

这些选项可通过任意组合来检测和分类所关注的包。规则选项使用逻辑“与”来处理，规则中所有的测试选项必须为真，这样规则才能产生正确的响应并调用程序执行规则的动作。

6.4.2　规则组成及含义

经过以上对 Snort 规则的介绍和应用举例，相信各位读者对于 Snort 的使用有了更深入的理解，当然，当我们把 Snort 作为 IDS 系统来使用时，需要详细了解这些过滤规则及组成和含义。多数 Snort 规则都是以单行进行描述，当然复杂的情况也有，规则因为长度的关系会搭配“/”来进行分行处理。例如：

```
   alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "ICMP ISS Ping";
itype;8;content: "ISSPNGRQ";     depth:32;
reference:arachnids,158;classtype:attempted-recon;sid:456; rev:4;)
```

Snort 的规则可分为两部分：

（1）Snort 规则头（Rule Header）

在规则头中包含了检测数据包的基本信息，一共包含了 7 个项目，每个项目都会记录着特定信息来提供给规则判断使用。

（2）规则动作（Rule Action）

这个项目包含下表 6-5 中列出的规则动作，用来针对数据包进行执行判断。

表 6-5　Snort 规则动作说明

规则动作	说 明
Alert	针对此数据包产生一个警报，并且记录
Log	将此数据包记录下来
Pass	忽略符合这个规则的数据包
Activate	针对此数据包产生一个警报，并且以另外一个 Dynamic 规则来进行数据包的处理
Dynamic	主要是针对 Activate 规则动作提供数据包处理时使用

Snort 规则中有 7 个重要参数需要说明。

（1）协议（Protocol）：常见的有 TCP、UDP 或 ICMP 协议等。

（2）源 IP（Source IP Address）：针对通过规则的数据包来源 IP 地址进行指定，可以通过正规表示式的方式来设定一个符合的源 IP 地址。如果在源 IP 地址前方添加上一个！号，则表示除此之外的 IP 地址数据包。例如：

```
alert tcp ! [192.168.150.0/24,192.168.200.0/24] any
```

在上面这个条件中，针对不是从 192.168.150.0/24 与 192.168.200.0/24 这两个网段的任一来源的 TCP 数据包进行警报。

（3）源端口（Source Port）：它可将规则定义的更细，常见的用法是使用 any 这个关键字来表示所有的端口，或是利用一段范围来表示一个指定的端口，下面简单举几个例子：

- 所有范围：log tcp 192.168.150.0/24 any
- 一段端口范围：log tcp 192.168.150.0/24 any 1:1024
- 某个特定端口：log tcp 192.168.150.0/24 80

除以上 3 种基本表示方式之外，还有下面三种特殊表示形式：

- 小于等于一个端口的表示：log tcp 192.168.150.0/24 :1024
- 大于等于一个端口的表示：log tcp 192.168.150.0/24 8000
- 排除某个端口的表示：log tcp 192.168.1.0/24 !23

（4）指示动作（Direction Operator）：通常是用于连接源 IP 与目标 IP 的关系，常见指示动作有以下两种：

①方向操作符号->：方向操作符号是常见的指示动作，符号的左边代表源 IP，符号的右边是数据包的目的 IP 及端口号，实际上是两条方向相反规则的一个合并，即源->目的，目的->源，在构建规则时，此规则将被拆分成为两条规则，例如：

```
log tcp any any ->192.168.150.0/24 443
```

这个含义是通过 Snort 来记录来自任何 IP 和端口的所有 TCP 数据包，只要这些数据包的目的地址是流向 192.168.150.0/24 这个网段中的 443 端口，便将这些数据包记录下来。

②双向操作符号 <>：它可以将源 IP、端口号和目的 IP 及端口号进行交换判断使用。例如：

```
log tcp 192.168.150.0/24 80 any <> 192.168.150.0/24 80
```

这句规则表示源 IP 可能是从 192.168.150.0/24 网段中的任何一个 TCP 数据包或是从 192.168.150.0/24 网段中 80 端口传过来的数据包，而目的地址除了为 192.168.150.0/24 网段中的 80 端口之外，也可能是 192.168.150.0/24 网段中任何一个地址，这些数据包都会被记录下来。

（5）目的 IP（Destination IP Address）：目的 IP 的使用方式与来源 IP 的使用方式相同。

（6）目的端口（Destination Port）：目的端口的使用方式与源端口的使用方式相同。

（7）规则选项（Rule Options）：刚才所介绍的规则头是 Snort 规则对于数据包进入时的最初对比，而比对匹配成功后的数据包怎么处理呢？应该执行哪些动作？这就是规则选项该发挥的作用了。

规则选项部分是整个 Snort 引擎的核心，使用规则选项时要特别注意两个符号的使用，分别是分号（;）与冒号（:）。Snort 所提供的规则选项符号很多。

这里主要介绍一般形态的规则选项（General Rule Options）。

在这个类型中的规则选项通常是用来提供资讯给规则进行使用，但这些资讯并不会影响到检测的结果或是过程。常见的规则选项如表 6-6 所示。

举例一：我们在 SIEM 中能看到 Snort 规则，这些规则的具体含义我们给大家讲解一下，先看图 6-22 中的例子。

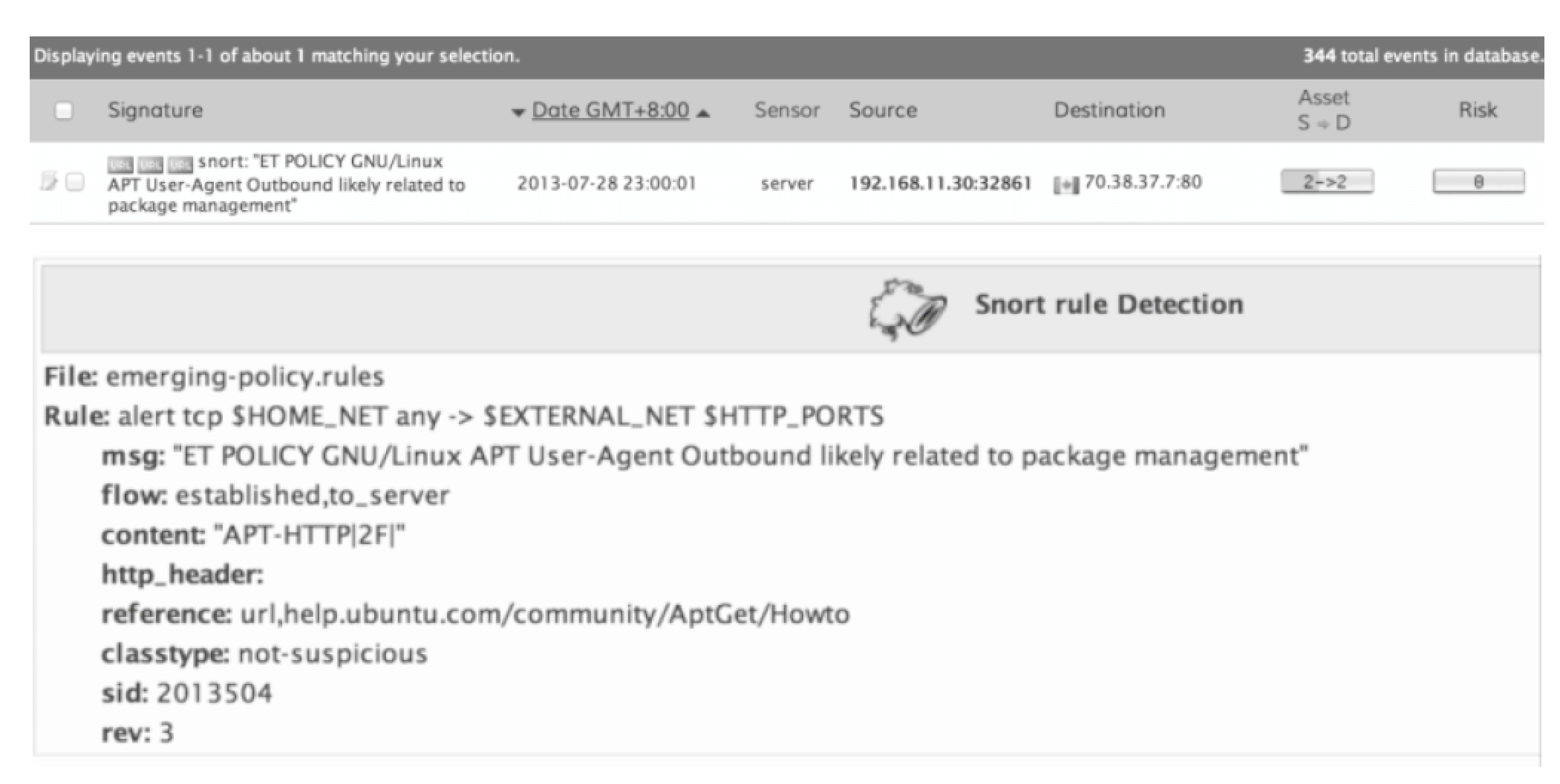


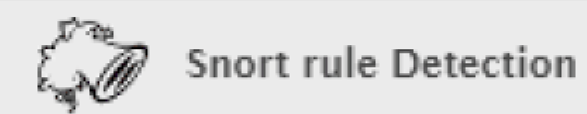
图 6-22 OSSIM 中 Snort 规则举例

规则解释如表 6-6 所示。

表 6-6　规则选项含义

规则选项	说 明
msg	msg 这个规则选项的主要功能是当封包检测符合规则时，在记录或是发出警报信息时，会将 msg 这个规则选项中所指定的文字信息一并记录或是在发出警报信息中显示。其语法格式为：msg：“<message text>”;图中显示为“ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management”
reference	Reference 此规则选项，主要是可以用来提供 Snort 使用其他的外部攻击识别系统来帮助入侵检测的强度，可以引入一些由 Snort 所认可的一些外部网站来进行使用，其语法格式为：reference:<id system>,<id>;
gid	有些 Snort 规则会显示 gid 这个规则选项, 它是指用来识别所设定的规则内容是属于 Snort 所规范的哪些事件，这些事件识别码都会预先定义在/etc/snort/gen-msg.map 文件（大小约 25KB）内。其语法格式为:gid:<generator id>;。预设规则如果不指定使用的 gid 识别码，那么是直接使用 1（snort general alert）来进行运作，读者若希望自定义新的 gid 识别码时最好是从一百万以后开始编码使用
sid	sid 这个规则选项是指用来识别所使用的 Snort 规则项目，这些 Snort 所使用的规则项目预先也会定义在/etc/snort/sid-msg.map 文件(大小约 2.7MB)内。其语法格式为:sid:<snort rules id>;。而 sid 所使用的范围基本上分为三大部分，分别为： • 小于 100:这是保留给未来扩充时使用 • 100～1,000,000：Snort 官方所预设配置的规则项目 • 大于 1,000,000:本机所自定义的 Snort 规则项目
classtype	classtype 这个规则选项可以用来将所涉及的入侵检测规则进行分类并且赋予一个预设的优先等级，Snort 提供了许多不同的规则分类，这些分类可以让 Snort 在运行时更有弹性。其语法格式为：classtype:<class name>;
rev	rev 这个规则选项是针对 Snort 规则所识别的修订识别码，一般在它上面都会搭配 sid。其语法格式为：rev:<revision integer>;
priority	可选项 priority 规则，它可以用来指定 Snort 规则的优先等级，其语法格式为: priority:< priority integer>;。如果读者在规则中使用 priority 时也搭配使用刚才所介绍的 classtype，由于 classtype 中有一个预设优先权选项，若是这个选项值也被启用时，将会覆盖 priority 规则选项中的值，读者在操作时需要注意
threshold	代表阈值，后面往往跟一个数值，例如：threshold:type threshold, count 5,seconds 60 表示 60s 内发生 5 次就告警。Track by_src 表示跟踪统计源 IP，track by_dst 表示统计目标 IP，而且是不同的 IP 到达阈值

举例二：同样是 emerging-plicy.rules 规则，我们看下面例子，如图 6-23 所示。



File: emerging-policy.rules
Rule: alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
    msg: "ET POLICY Outdated Windows Flash Version IE"
    flow: established,to_server
    content: "x-flash-version|3a| "
    http_header:
    content: !"11,8,800,94|0d 0a|"
    distance: 0
    within: 13
    http_header:
    content: "MSIE "
    http_header:
    pcre: "/^User-Agent\x3a[^\r\n]+?MSIE/Hm"
    threshold: type limit, count 1, seconds 60, track by_src
    reference: url,www.adobe.com/software/flash/about/
    classtype: policy-violation
    sid: 2014726
    rev: 21

图 6-23 OSSIM 中 Snort 规则举例二

Classtype 含义如表 6-7 所示。

表 6-7 Snort 中 classtype 含义

classtype	含义	优先级
Attempted-admin	试图取得管理员权限	高
Successful-admin	已取得管理员权限	高
Policy-violation	潜在隐私入侵	高
Shellcode-detect	发现 shellcode 攻击	高
Trojan-activity	发现木马攻击	高
Web-application-attack	Web 应用程序攻击	高
Attempted-dos	拒绝服务攻击	高

Reference 所支持的外部系统的网址如表 6-8 所示。

表 6-8 Reference 所支持的外部系统

支持系统 ID	网址
Bugtraq	http://www.securityfocus.com/bid
nessus	http://www.nessus.org/plugins/
cve	http://cve.mitre.org/compatible/
osvdb	http://osvdb.org/show/osvdb/
arachNIDS	http://www.whitehats.com/info/IDS

在 OSSIM 中这些网址在/etc/snort/reference.config 文件中定义，然后会被 Snort 主配置文

件 snort.conf 所引用。Payload（有效载荷）的规则选项含义如表 6-9 所示，通常可以针对特定的数据包内容来设定搜索条件。

表 6-9 Payload 规则选项含义

规则选项	含义
content	这个选项可允许使用者自定义关键字，进行数据包内容的有效载荷搜索。语法为：content: [!] “<content string>”，它可以搭配关键词来搜索，具体关键词如表 6-10 所示
pcre	这一规则选项允许使用在涉及 Snort 规则时使用 Perl 相兼容的表达式
Uricontent	这个规则代表允许使用者通过正规表达式处理 URI 内容，语法为：uricontent: [!]”<content string>”

content 选项中关键词的使用如表 6-10 所示。

表 6-10 content 选项中关键词的使用

关键词	含 义
Offset	它用来指定 Snort 规则在搜索数据包要从哪一个地方开始搜索，语法为：offset:<number>，例如 alert tcp any any -> any 8080（content:”cgi-bin/open”; offset 10;depth:6;）这条规则含义是针对传入到 8080 端口的 tcp 数据包，跳过前面 10 个单位内容，往后 5 个单位内容搜索 cgi-bin/open 的字符串
Distance	指定 Snort 规则在搜索数据包内容时可以用来确认搜索的内容
Within	指定 Snort 规则在搜索数据包内容
http_header	用来指定 Snort 规则在搜索数据包时, 是以 HTTP 用户端连接请求的数据包的 HTTP 表头内容进行搜索。与它的使用含义类似的关键词还有 http_cookie、http_method、http_uri 等

Snort 规则的编写能力非一日之功，在平时大家多实战，多总结，这里介绍一个小窍门，利用网络分析光盘（BT4/5、DEFT 8）中蕴藏着丰富的测试工具，通过渗透测试和模拟攻击来快速获得比较全面的信息，另外大家还可以参考这个 Snort 在线规则生成器来学习，http://183.62.34.17:8080/1080p/asp/snort/snortrules.html 效果如图 6-24 所示。

图 6-24 在线规则生成

6.4.3　编写 Snort 规则

编写规则之前，一定要注意语法并注意逻辑关系，违反语法的 Snort 规则将不能被载入到检测机制中，不合逻辑的规则无法生效。如果载入语法有错的规则，那么可能导致不可预料的后果。

（1）基础知识

对于刚接触 Snort 规则编写的新手而言，操作 Snort 规则最简单的一种方法就是对已有的规则进行修改。假设只有一台 IIS 服务器（IP=192.168.1.1），想修改与 IIS 相关的规则使它们仅仅应用在这台服务器上，而不是用在每台 Web 服务器上。起始可能想要修改 Snort-sigs 邮件列表中后缀为“.htr chunked”的编码规则，内容如下：

```
   alert tcp $EXTERNAL_NET any -> $ HTTP_SERVERS $ HTTP_PORTS (msg:"WEB-IIS.htr
chunked encoding"; uricontent:".htr";classtype:web-application-attack;rev: 1;)
```

为了使它仅仅应用于 IIS 服务器上，应改为：

```
   alert tcp $EXTEBNAL_NET any -> 192.168.1.1 $ HTTP_PORTS (msg:"WEB-IIS.htr
chunked encoding"; urioontent:".htr";classtype: web-application-attack;rev: 2;)
```

现在可以在 192.168.1.1 的 Web 服务器上使用该规则，注意“rev”关键字由 1 变为 2，表明这是一个已存在规则的新版本。

下面进一步提炼该规则，希望仅在向服务器提出请求时使用该规则，因为该方向的流量可能是某种攻击。也可能希望仅在已建立的 TCP 会话中应用该规则，阻止攻击者利用误报的洪水进行 DOSing Snort 攻击。可以加入 flow 选项，例如：

```
   alert tcp $ EXTERNAL_NET any -> 192.168.1.1 $ HTTP_PORTS(msg:"WEB-IIS .htr
chunked encoding"flow:to_server,established;uricontent:".htr"; classtype:
web-application-attack; rev:3;)
```

加载该规则之后，删除误报数量减少。

（2）技能提高

当网站允许恶意代码被插入到一个动态创建的网页中时，跨站脚本（XSS）攻击就会发生。如果不能正确地检查用户输入，攻击者就可以在网页中嵌入脚本，这些脚本使 Web 应用程序不能按照预期执行。大多数的 XSS 攻击需要向特定页面请求中插入脚本标记。可以使用 XSS 攻击的这个特征编写规则。因为只要向 Web 应用程序插入 XSS 脚本，就会使用到<SCRIPT>、<OBJECT>、<APPLET>和<EMBED>等标记。例如，当发现<SCRIPT>标记时，触发该规则。首先，应该创建一个规则，触发包含“<SCRIPT>”字符串内容的流量：

```
   alert tcp any any -> any any (content:"<SCRIPT>"; msg:"WEB-MISC XSS attempt";)
```

XSS 攻击会触发该规则，与此同时许多其他的正常流量也会触发这个规则。例如，假设某人发送一个嵌有 JavaScript 代码的电子邮件，Snort 也会发出报警，从而产生误报。为了避

免这种情况的发生，就需继续修改这个规则，使其仅在 Web 流量中触发：

```
alert tcp $EXTERNAL_NET any ->$ HTTP_SERVERS $ HTTP_PORTS (content:
"<SCRIPT>";msg:"WEB-MISC XSS attempt";)
```

现在，仅在来自 Web 服务器的相关 HTTP 会话中检测到<SCRIPT>内容时，才会触发该规则。当流量开始于一个外部的 IP 地址（$EXTERNAL_NET），并被发送给 Web 服务器（$ HTTP_SERVERS）上 HTTP 服务端口（$HTTP_PORTS）时，该规则才被触发。但在载入这个规则之后，会发现无论何时有包含 JavaScript 的请求时，还会产生大量的误报。因此，需要更进一步地改进该规则，找到 XSS 流量的唯一特征。

当客户在请求中嵌入<SCRIPT>标记时会发生 XSS。如果服务器发送请求响应的<SCRIPT>标记，它可能是正确的流量（如 JavaScript），而不是一个 XSS 攻击。可以使用这个 XSS 攻击特征进一步提炼该规则：

```
alert tcp $ EXTERNAL_NET any-> $HTTP_SERVERS $ HTTP_PORTS (msg:"WEB-MISC XSS
attempt"; flow:to_server,established;content:"<SCRIPT>";)
```

这个改进后的规则使用了 flow 选项，该选项使用 Snort 的 TCP 重建特征来鉴别流量的方向。通过应用特定的 flow 选项，“to_server”和“established”规则仅对从客户端向服务器端发起的会话有效。XSS 攻击只有可能发生在这个方向传输的流量中，而反方向上的流量则可能是一个包含 JavaScript 标记的正常的 HTTP 会话。

现在规则已经可以识别 XSS 攻击了，接着需要利用大小写敏感性确保攻击者不能躲避规则。Contend 选项可用来区分大小写，然而 HTML 不区分大小写，因此攻击者可以通过将脚本标记修改为< script>或<script >避开这个规则，为了弥补这一点，应用 contend 选项来指定不区分大小写。

```
alert tcp $ EXTERNAL_NET any-> $ HTTP_SERVERS $ HTTP_PORTS (msg:"WEB-MISC XSS
attempt"; flow:to_server,established; content:"<SCRIPT>"; nocase;)
```

为了完成该规则，还需给它赋予一个较高的优先级。

```
alert tcp $ EXTERNAL_NET any ->$ HTTP_SERVERS $ HTTP_PORTS (msg:"WEB-MISC XSS
attempt";flow:to_server,established; content:"<SCRIPT>"; nocase;priority:1;)
```

由于 Snort 不支持对主机名的解析，所以 IP 地址只能使用 CIDR 的形式。

在规则中，可以使用否定操作符对 IP 地址进行操作，表示 Snort 除了列出的 IP 地址外，匹配所有的 IP 地址。否定操作符使用!表示。

例如：

```
alert tcp !192.168.1.0/24  any -> 192.168.150.0/24 111(content:"|00 01 86
a5|";msg:"externalmountd access"
```

（3）端口说明

在规则中，可以有几种方式来指定端口号，包括：any、静态端口号定义、端口范围。any 表示任意合法的端口号；静态端口号表示单个的端口号，例如：111（portmapper）、23（telnet）、80（http）、443（https）等。使用范围操作符:可以指定端口号范围。有几种方式来使用范围操作符:达到不同的目的。

例如：

```
log udp any any ->192.168.150.0/24  1:1024
```

记录来自任何端口，其目的端口号在 1~1024 的 UDP 数据包。

```
log tcp any any ->192.168.150.0/24 :600
```

记录来自任何端口，其目的端口号≤600 的 TCP 数据包。

```
log tcp any :1024 ->192.168.1.0/24  500:
```

记录源端口≤1024，目的端口≥500 的 TCP 数据包。

（4）方向操作符

方向操作符->表示数据包的流向。它左边是数据包的源地址和源端口，右边是目的地址和端口。此外，还有一个双向操作符<>，它使用 Snort 对这条规则中两个 IP 地址/端口之间双向的数据传输进行记录/分析，例如 telnet 对话，下面的规则表示对一个 telnet 对话的双向数据传输进行记录：

```
log !192.168.150.0/24 any <> 192.168.150.0/24 23
```

（5）实战

Snort 的运行效果取决于规则，使用不精确或误报的规则对于网络安全威胁很大。如果想为某种特殊的攻击编写 Snort 规则，则需要在测试环境中重现攻击过程，并且保证嗅探器能够监听到通信，然后捕获从攻击机发出的包以及被成功攻击的受害机发出的应答包。

例如，BT 下载消耗网络资源，但 BT 下载过程中，种子列表是动态变化的，不可能通过在 Iptables 防火墙汇总添加固定的规则，起到限制 BT 下载的目的，我们在 Snort 中添加规则，来达到阻断 BT 下载的目的。

分析过程：首先通过抓包工具如 Wireshare，抓取 BT 下载的数据流作为研究样本。经过分析可知 BT 客户端向服务器请求种子列表的 GET 报文中一般包含如下典型特征内容：

```
"GET", "/announce", "info_hash","event=started";
```

而 BT 客户端和种子列表开始交互数据包中包含如下特征：

```
"|13|BitTorrent Protocol"。
```

基于这两个特征我们编写两条规则，然后添加到规则库中。

- 规则 1

```
alert tcp $HOME_NET any ->$EXTERNAL_NET any(msg:"P2P BitTorrent announce request";flow:to_server,established;content:"GET";depth:4;content:"/announce";distance:1;content:"info_hash=";offset:4;content:"event=started";offset:4;classtype:policy-violation;sid:2780;rev:3;)
```

该规则用来匹配包含"GET"、"/announce"、"info_hash="、"event=started"内容的 TCP 数据包文，它们之间是“与”的关系。

- 规则 2

```
alert tcp $HOME_NET any ->$EXTERNAL_NET any(msg:"P2P BitTorrent transer";flow:to_server,established;content:"|13|BitTorrentprotocol";depth:20;clastype:policy-violation;sid:2780;rev:3;)
```

此规则用来匹配包含“|13|BitTorrentprotocol”的客户端，向服务器发送的请求种子列表的报文及 BT 客户端之间交互的 BitTorrent Protocol，然后发出报警信息。

（6）规则统计 ID

Snort 中每条规则都对应唯一的规则 ID 号：sid，在 CVE 中对每个规则相对应的漏洞，都有一个相应的漏洞描述文件，该文件是以漏洞 ID 号命名的，里面描述了该漏洞的概要（Summary）、详细信息（Detailed Information）、受影响的系统（Affected Systems）等。为了方便统计，我们把所有的描述文件导入数据库，再通过 sid 关联把规则文件（rules）的相应字段导入到数据库中，然后在数据库中通过关键字进行统计。

6.4.4 手工修改 Suricata 规则

Suricata 规则库路径在/etc/suricata/suricata.yaml 中定义，它们在/etc/suricata/rules 目录下，根据分类去修改相关文件，例如要添加 Nmap 扫描的规则，则编辑 emerging-scan.rules 文件。在 OSSIM 中如何修改规则。

（1）首先编辑规则

```
#vi /etc/snort/rules/my.rules
alert tcp any any -> any 112 (msg:"TCP Traffic";)
```

（2）把 my.rules 规则添加到 snort.conf 中。

编辑/etc/snort/snort.conf，在文件末尾添加：

```
#添加新规则
include $RULE_PATH/emerging-my.rules
```

6.4.5 启用新建的 ET 规则

在 Snort 中所谓“ET 规则”代表 Emerging Threats，它可以在下面网址下载。

```
http://rules.emergingthreats.net/open-nogpl/snort-2.9.0/emerging.rules.tar.
```

```
gz
```

下面分几个步骤讲解如何安装这个 ET 规则：

（1）首先下载、解包。例如解到/tmp/rules 目录中。

（2）备份旧的规则：

```
#cp  /etc/snort/rules /opt/rulesbackup/   \\*这里的 rulesbackup 自己定义，备份目录
```

（3）将新规则复制到下面目录：

```
#cp /tmp/rules/*  /etc/snort/rules
```

（4）更新#OSSIM Server：

```
#cd /usr/share/ossim/scripts
#perl create_sidmap.pl /etc/snort/rules
Loading from reference_system…done
Loading from references … done
……
Insert into sig_reference … done
```

（5）重配置 OSSIM：

```
#ossim-reconfig
```

重新配置后，新的 ET 规则即可生效。

6.4.6　应用新规则

读者在初次使用时，在系统/etc/snort/rules/local.rules 加了条规则，为何不生效呢？

比如使用 Snort 检测可执行程序（.exe）添加如下规则：

```
    alert tcp any any -> any any (msg: "DLL Windows file download"; flow: established;
content:"MZ";isdataat: 76,relative;content:"This program cannot be run in DOS
mode."; distance: 0; isdataat: 10,relative; content:"PE"; distance: 0; classtype:
misc-activity; sid:5000789;)
```

在/etc/snort/rules/local.rules 中添加规则后，还需要执行如下命令：

```
#perl  /usr/share/ossim/scripts/create_sidmap.pl  /etc/snort/rules/
```

注意：sid-msg.map 和 gen-msg.map 的区别：

- /etc/snort/gen-msg.map：gid 对应的报警文本。
- /etc/snort/sid-msg.map：sid 对应的报警文本，包含一个从 msg 标签到 snort 规则 ID 的映射。

注意，最后需要重启 Snort 服务。

6.4.7 主动探测与被动探测

现有的网络探测技术可以分为两类：主动探测和被动探测。主动探测通过主动扫描来检查监听服务的存在，其优点是扫描快速和结果完整，它的缺点是具有攻击性质（大规模的并行高速扫描会对系统和网络造成的严重影响），易暴露，易受防火墙影响，只能提供网络当前的快照，当然网络组成或结构发生变化需要重新扫描。例如 Nmap 工具就是主动探测工具。

被动探测一般通过网络嗅探来提取服务端的相关信息，被动探测结果不会受防火墙影响，而且被动探测完全没有入侵性。被动探测的缺点是检测的服务不能及时发现，某些网络服务可能要花数天时间才能发现，例如 arpwatch（MAC 地址异常检测）p0f（被动 OS 检测）、pads（服务异常检测）等属于被动检测工具，它们在 OSSIM 3.1 中都能找到，稍后在第 8 章将深入研究。

6.5 可疑流量检测技术

在网络流量分析中，较为有效的检测是基于特征码的检测，因为恶意流量（如病毒、僵尸网络、木马、蠕虫、Rootkit 等）在构造和内容上都不同于正常流量，所以我们可以创建攻击特征来进行比对检测。

6.5.1 通过特征检测

以 OSSIM 系统中的 Snort 为例，一个恶意流量的特征可以被创建成一个规则，而载入 Snort 检测引擎（负责特征匹配）。我们回顾一下，Snort 是如何对规则进行处理，重点要了解 Snort 检测引擎如何工作，下面看个例子：

```
   alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING
NMAP";dsize:0;type:8;)
```

该规则表示：对于来自外部网络、有效载荷数据为空，并且 ICMP 类型值为 8 的 ICMP 流量产生报警。

为了便于大家理解，这里假设一个场景：攻击者利用 Nmap 工具对一个网段进行扫描，这里利用 nmap ping 的一个特点，它将 ICMP 类型域值设定为 8，并且有效载荷为空。攻击者发送这样的数据包，发现了该网段存活的主机，进而再对其进行更深入的扫描。

由于 Snort 监控的端口在交换机上做了端口镜像，那么在该网段发生的所有扫描都会被记录下来。

6.5.2 检测可疑的载荷

Snort 特征并不只是检测 IP 包头的数据，还能进行深度检测。我们需要检测隐藏在一个看

似正常封包中的可疑载荷，例如：在 OSSIM 4 系统中，打开/etc/snort/rules/deleted.rules 文件，查看第 5807 行的规则，如图 6-25 所示。

```
5807  alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"DELETED NETBIOS SMB IPC$ unicode share acce
     ss"; flow:established,to_server; content:"|00|"; depth:1; content:"|FF|SMBu"; within:5; distanc
     e:3; byte_test:1,&,128,6,relative; pcre:"/^.{27}/sR"; byte_jump:2,7,little,relative; content:"I
     |00|P|00|C|00 24 00 00 00|"; distance:2; nocase; flowbits:set,smb.tree.connect.ipc; flowbits:no
     alert; classtype:protocol-command-decode; sid:538; rev:17;)
```

图 6-25　一段 snort 规则

这条规则的含义为，对任何一个净载荷中含有“I|00|P|00|C|00 24 00 00 00|”，并且包头表明是从外部网络发给运行 SMB（Server Message Block）服务的计算机的 TCP 流量产生报警。在检测中发现，sid=538 所占比例比较大，这种净载荷可能会引起 Windows 系统缓冲区溢出，最终机器崩溃。只有这种净载荷是从外部发给运行 NETBIOS（TCP139 端口）的计算机时，才需要注意这一规则。

SMB 协议基于 TCP-NETBIOS，NetBIOS 相关的端口分别是：137、138、139。其中，137 端口表示 NETBIOS Name Service，138 端口表示 NETBIOS Datagram Service，139 端口表示 NETBIOS Session Service。

6.5.3　检测具体元素

从系统性能上考虑，Snort 特征可以具体针对特殊协议的某一个元素描述进行检测，下面看个例子：

查看/etc/snort/rules/web-iis.rules 文件中第 136 行内容，如图 6-26 所示。

```
136 alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI .ida attempt"; flow
    :to_server,established; content:".ida?"; fast_pattern; nocase; http_uri; metadata:service http;
    reference:arachnids,552; reference:bugtraq,1065; reference:cve,2000-0071; reference:cve,2001-050
    0; classtype:web-application-attack; sid:1243; rev:19;)
```

图 6-26　删除 snort 规则

该规则代表对任何一个来自外部网络发往本地 Web 服务器，并且 URL 中包含“.ida?”的网络流量产生报警。

Microsoft IIS .IDA / .IDQ ISAPI 扩展远程路径泄露漏洞出现在 Windows NT/2000 的系统中，是个古老的漏洞了。.ida 扩展名是 Windows NT/2000 系统索引服务的组件，通过 IIS 的.ida、.idq 映射可以使用 Index Server 检索，IIS 对于不存在的.ida、.idq、文件的访问请求没有很好地处理。可能导致 WEB 服务器泄露网站文件的绝对路径。

比如，对于未修复 MS01-033 补丁的一台 Windows 2000 Server IIS 服务器，就会遭受 CodeRed.F（红色代码）病毒的攻击，并向其他未打补丁的机器传播，具体现象是发出大量“GET /default.ida?XXXX”请求，那么这种含有.ida 的流量就会被 Snort 发现并报警。对于这类漏洞，最新的 Windows 系统已消除。

6.5.4 OSSIM 中的 Snort 规则与 SPADE 检测

在 OSSIM 系统里的 Snort 软件集成了大约 9000 种规则，看似覆盖面很广，但这是通用规则，有很多规则需要根据网络情况进行“私人定制”，OSSIM 平台提供了二次开发的环境，所有规则都可以增、删、改。

OSSIM 3 系统中提供了 SPADE（Statistics Packet Anomaly Detect Engine，统计包异常检测引擎），异常检测的手段可通过 Arpwatch 检测 MAC spoofing，通过 P0f 检测操作系统变化，通过 Pads 和 Nmap 检测新增网络服务，用它还可以检测无匹配特征的一些可疑流量。具体原理是，SPADE 观测网络流量，并建议一张描述网络正常时的流量表作为基准。OSSIM 发展到 5.x 版后用 prads 代替原先 arpwatch 和 pads 的功能。

这张表记载的数据包的类型、源地址、目的地址，当达到一定大小后，SPADE 挑出每个包将被赋值，该数值的大小取决它在表中出现的频率，频率越高，该值越小。当该数值达到某一配置阈值时，就会产生报警。

这种方法对于检测黑客的异常行为非常奏效，有些攻击者会缓慢地扫描服务器端口，企图把自己的扫描数据，淹没在大量合法数据包中，以达到隐蔽目的，即使是使用了多个源地址进行活动，也会被 SPADE 察觉。又如，假定我们使用 SPADE 功能保护一台 WEB 服务器。首先我们把 Web 服务器所有流量镜像到交换机 SPAN 口，我们的 OSSIM 系统在哪里进行监听呢？

SPADE 为进入服务器的流量建立一张表，这些流量大部分是对 80 和 443 端口的 TCP 链接，建立后，对 80 和 443 端口的 TCP 请求就是正常流量，这时会被赋予一个较低的数值。如果这时攻击者想要查明该服务器有哪些开放端口，SPADE 就会给这些流量赋予一个较高的值，因为它对于这台服务器是比较少见的。此时，如果该值超过了事先定义好的阈值，系统就会发出报警。

6.5.5 恶意代码行为特征分析

随着恶意代码技术的发展，涌现出了各种恶意代码分析技术。当前主流的恶意代码分析技术是静态分析等技术，分析的目的就是依靠工具找到特征码。

比如，在 2004 年 3 月发现的一种蠕虫，名叫 W32.Netsky.P@mm，它是一种群发邮件蠕虫，它使用自己的 SMTP 引擎将自己发送到在扫描硬盘和映射的驱动器时找到的电子邮箱。此蠕虫还试图利用各种文件共享程序、通过将自己复制到各个共享文件夹进行传播，在一台受感染的机器上，我们找到了该蠕虫的样本，如图 6-27 所示，并对其二进制文件进行了分析。

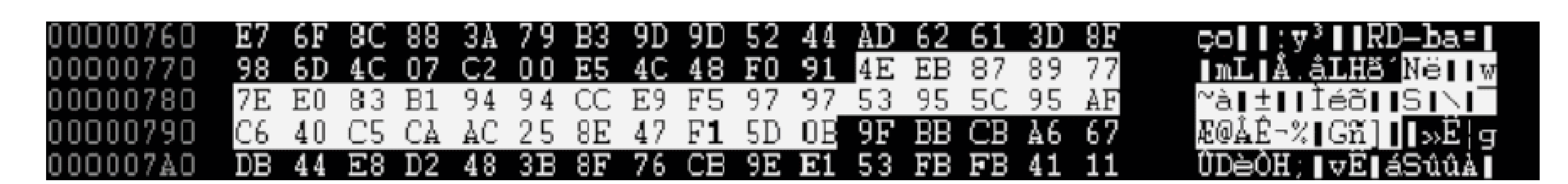


图 6-27 蠕虫样本

当我们找到蠕虫特征码以后就很容易写出检测规则，详情如下：

```
alert tcp $EXTERNAL_NET any -> any any (msg:"W32.NetSky.p@mm - SMB";content:"|4E
```

```
EB 87 89 77 7E E0 83 B1 94 94 CC E9 F5 97 97 53 95 5C 95 AF C6 40 C5 CA AC 25 8E
47 F1 5D 0E|"); classtype:misc-activity;rev:1;)
```

6.5.6　蜜罐检测

对于蠕虫样本的提取，主要采用的方式是基于蜜罐系统捕获。在这种可控的模拟环境中，可以消耗黑客精力，并记录黑客来源和犯罪证据。另外，IDS 的特征库需要不断地更新，才能检测到新的攻击行为。

蜜罐系统将记录到的入侵信息传递给入侵检测系统，入侵检测系统收到事件信息后，对其进行分析并提取攻击特征，最后将新的攻击特征添加到系统的攻击特征库中，从而使得入侵检测系统可以检测出未知的攻击手段。利用蜜罐系统与入侵检测系统的联动，可以增强入侵检测系统对网络的防御能力，这个系统的架构如图 6-28 所示。

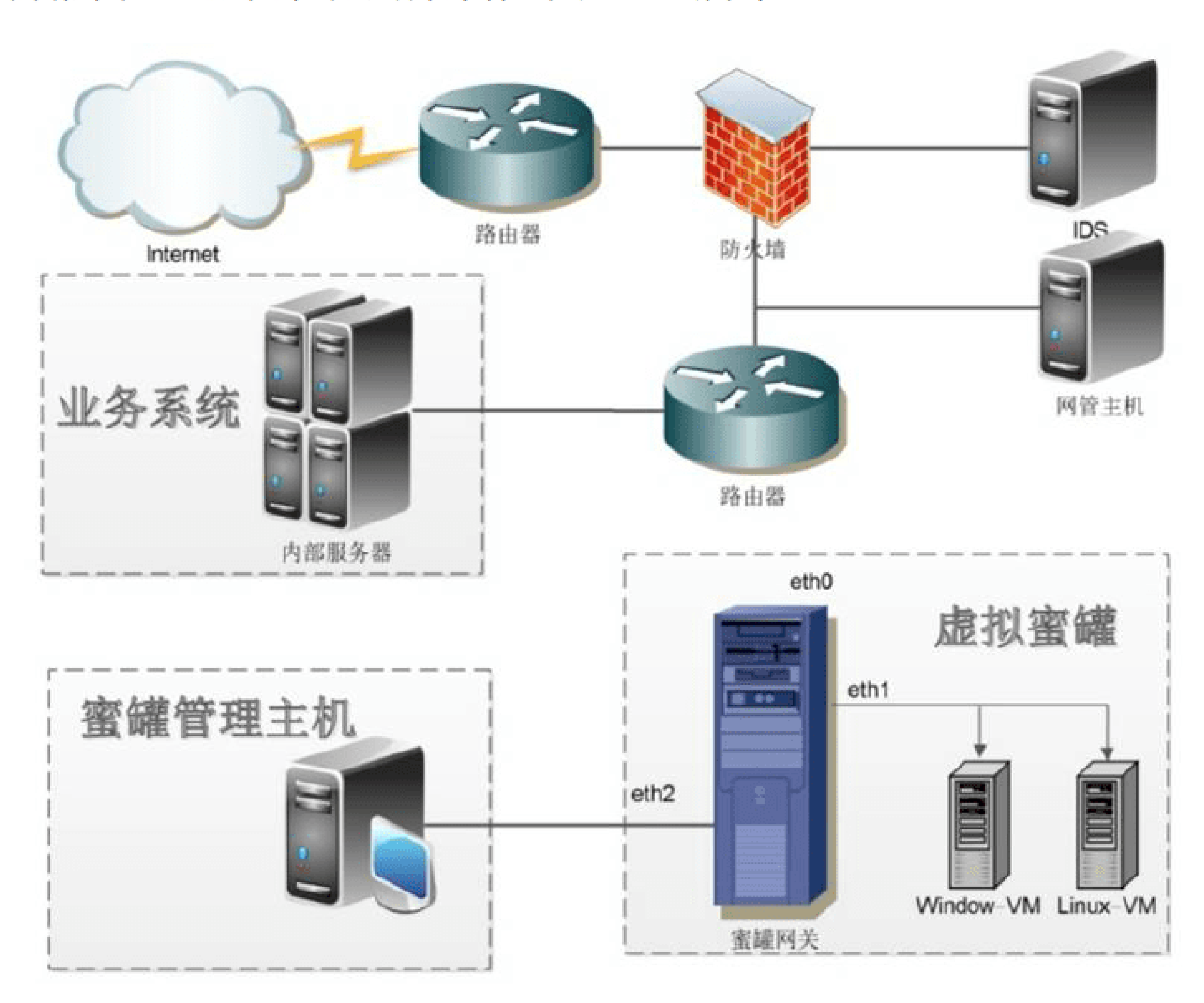


图 6-28　虚拟蜜罐架构图

OSSIM 系统中的 Snort 软件支持对报警定制，并支持优先级。可以根据需要添加许多自定义的报警分类。就像特征码一样，报警分类也是通过规则来定义。比如，一个针对特洛伊木马流量的报警分类如下：

```
config classification:trojan-activity . A Network Trojan was detected
```

大家可以参考/etc/suricata/rules/emerging-trojan.rules 规则文件第 3212 行:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET TROJAN Chorns/Poison Ivy
related Backdoor Ini     tial Connection"; flow:established; dsize:12;
content:"/FIRSTINF/|0d0a|"; reference:url,doc.emergin     gthreats.net/2010344;
classtype:trojan-activity; sid:2010344; rev:3;)
```

该分类针对所有已检测到的特洛伊木马行为，包括 Netbus、Subseven 等。这一规则规定将严重的定义为 1，可以增加这个值来降低严重级别。比如，在特征规则中，Subsenven 特征中就带有 trojan-activity 来表示，这样的例子还包括/etc/suricata/rules/emerging_pro-trojan.rules 规则文件的第 12255 行：

```
   alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ETPRO TROJAN
Trojan.SubSeven.215-srv reporting via ICQ WWW script"; flow:established,to_server;
content:"/scripts     /WWPMsg.dll?from="; http_uri; content:"(Win95|3b|";
fast_pattern; http_user_agent; content:"&subject=Pager&body="; http_uri;
reference:md5,68fac0fc380614f4eb08c8485e     a876c4; classtype:trojan-activity;
sid:2805959; rev:4;)
```

假设忽略 subseven 的影响，那么可以通过降低优先级的方法解决，实现起来很简单，可以用一个优先级标识符 priority:2。

还是接着上面的例子修改内容为：

```
   alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ETPRO TROJAN
Trojan.SubSeven.215-srv reporting via ICQ WWW script"; flow:established,to_server;
content:"/scripts     /WWPMsg.dll?from="; http_uri; content:"(Win95|3b|";
fast_pattern; http_user_agent; content:"&subject=Pager&body="; http_uri;
reference:md5,68fac0fc380614f4eb08c8485e     a876c4; priority:2; sid:2805959;
rev:4;)
```

最终通过这种方法可以使报警级别达到最合理的状况。

6.6 Snort 规则进阶

前面几节介绍了 Snort 规则基础，下面讲解 Snort 规则集的另外一些实用功能。

6.6.1 可疑流量的报警

假设一个场景：在内网中，用 Snort 监控的 VLAN 中的网络交换机，通过 Telnet 来管理它们。在这种情况下，这条 Snort 规则所发出的警报就算是正常。

我们打开 OSSIM 系统中/etc/suricata/rules/emerging-telnet.rules 配置文件，查看第 53 行，如图 6-29 所示。

```
53 alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"GPL TELNET TELNET access"; flow:from_ser
   ver,established; content:"|FF FD|"; rawbytes; content:"|FF FD|"; distance:0; rawbytes; content:"
   |FF FD|"; distance:0; rawbytes; reference:arachnids,08; reference:cve,1999-0619; reference:nessu
   s,10280; classtype:not-suspicious; sid:2100716; rev:14;)
```

图 6-29　snort 规则

如果把这条规则用在了监控一个受禁止使用 Telnet 进行访问的防火墙上，那么它发出的报警

就非常重要，应该引起管理员的注意。就是说一定要注意产生报警的具体环境，而不能一概而论。

下面再看个例子，Web 中的一些敏感资源不让访问，若有人尝试访问 Web 服务器所禁止访问的资源时，就会由下面这条规则所触发报警。

打开 emerging-misc.rules 配置文件，查看第 85 行，如图 6-30 所示。

```
85 alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:"GPL MISC Connection Closed MSG from Port 80";
   flow:from_server,established; content:"Connection closed by foreign host"; nocase; classtype:unk
   nown; sid:2100488; rev:5;)
```

图 6-30 snort 规则

6.6.2 空会话攻击漏洞报警

早在 Windows 2000 系统时，曾发现空会话攻击漏洞（当然现在 Windows 7/8 系统已经消除此问题），攻击者可以很轻松地利用空会话完成攻击，通过在主机上面建立一个空会话，如图 6-31 所示，再通过 Enum 枚举共享等方式，攻击者可能会获得一个重命名管理员的账户，这是因为这个空会话会加载到所有该计算机账户组中。它使得不在域中的主机也能使用该主机所使用的网络资源。

```
C:\WINNT\system32\cmd.exe
C:\>
C:\>net use \\127.0.0.1\ipc$ "" /u:""
The command completed successfully.
```

图 6-31 建立空会话

图 6-31 使用 Net 命令发起的成功空会话连接，当使用 Snort 监控这些主机时，如有人再利用这种方式攻击，就会立刻暴露出来。在 OSSIM 系统中，先打开 emerging-netbios.rules 文件，查看 294 行，Netbios Null 会话的规则如图 6-32 所示：

```
294 alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"GPL NETBIOS NT NULL session"; flow:to_server
    ,established; content:"|00 00 00 00|W|00|i|00|n|00|d|00|o|00|w|00|s|00| |00|N|00|T|00| |00|1|00
    |3|00|8|00|1"; reference:arachnids,204; reference:bugtraq,1163; reference:cve,2000-0347; classt
    ype:attempted-recon; sid:2100530; rev:11;)
```

图 6-32 snort 规则

当攻击者企图通过匿名方式，连接枚举用户或其他系统信息时，该规则将会被触发。

6.6.3 用户权限获取

拥有权限使用一个普通用户账号的攻击者，可以利用系统的各种漏洞来提升自己的权限，直到获取系统最高权限。比如，可以通过修改.rhosts 文件来获取全局访问权限。下面是 Snort 规则检测用户权限获取的企图行为：打开 emerging-misc.rules，查看 139 行，如图 6-33 所示。

```
139 alert tcp $EXTERNAL_NET any -> $HOME_NET 514 (msg:"GPL MISC rsh echo + +"; flow:to_server,establ
    ished; content:"echo |22|+ +|22|"; reference:arachnids,388; classtype:attempted-user; sid:210060
    8; rev:6;)
```

图 6-33 snort 规则

在实际应用中，提权的目标很可能是数据库，用户可以访问一种类型的数据，但希望访问其他受限制的资源，另外，如果用户获得超级用户权限，它就可以控制主机。在 Snort 中监控

这种用户提权的例子是 Microsoft SQL Server 的 xp_cmd shell 规则，打开 emerging-exploit.rules，查看 931 行，如图 6-34 所示。

```
931 alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 1433 (msg:"GPL EXPLOIT xp_cmdshell - program execut
    ion"; flow:to_server,established; content:"x|00|p|00|_|00|c|00|m|00|d|00|s|00|h|00|e|00|l|00|l|
    00|"; nocase; classtype:attempted-user; sid:2100687; rev:6;)
```

图 6-34　snort 规则

Xp_cmdshell 可以用于执行 SQL Server 的系统命令。攻击者可使用 xp_cmdshell 安装木马或执行可以在 Windows 命令行下执行的任何程序，该规则就能检测这种攻击行为。

6.6.4　失败的权限提升报警规则

攻击者提升用户权限也不是一帆风顺，在提权过程中，失败的登录行为就会触发系统大量报警，这样攻击者的行为就会被发现。又如 PCAnywhere 失败登录规则就是一个用户权限获取的例子，我们打开 emerging-policy.rules 文件，查看 1706 行，如图 6-35 所示。

```
1706 alert tcp $HOME_NET 5631:5632 -> $EXTERNAL_NET any (msg:"GPL POLICY PCAnywhere Failed Login"; f
    low:from_server,established; content:"Invalid login"; depth:16; reference:arachnids,240; classt
    ype:unsuccessful-user; sid:2100512; rev:5;)
```

图 6-35　snort 规则

该规则用于检测登录 PCAnywhere 失败的尝试。

6.6.5　企图获取管理员权限

如果攻击者可以访问 Windows 系统里的 Admin$共享资源，它就可以访问 Windows 系统目录（c:\winnt\），但这么做的人，并不是真正的管理员，怎么发现问题呢？下面这条规则就可以检测这种行为，打开 emerging-netbios.rules 文件，查看第 303 行，如图 6-36 所示。

```
303 alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"GPL NETBIOS SMB ADMIN$ share access"; flow:e
    stablished,to_server; content:"|00|"; depth:1; content:"|FF|SMBu"; within:5; distance:3; byte_t
    est:1,!&,128,6,relative; byte_jump:2,34,little,relative; content:"ADMIN|24 00|"; distance:2; no
    case; classtype:protocol-command-decode; sid:2100532; rev:14;)
```

图 6-36　snort 规则

6.6.6　成功获取管理员权限

Unix 系统的 passwd 文件是不能在 TFTP 这样不安全协议上传送的，TFTP 协议既不需要认证，也不加密，极度危险。所以在老式 Unix 系统中，可以通过下面方式获取 passwd：

```
$tftp 1.1.1.1
tftp>connect 1.1.1.1
tftp>get /etc/passwd /tmp/passwd.cracker
tftp>quit
```

通过 Snort 就能检测到这种攻击，下面这条是通过 TFTP 传输 passwd 文件的行为。

打开 emerging-tftp.rules，查看 89 行规则，如图 6-37 所示。

```
89 alert udp any any -> any 69 (msg:"GPL TFTP GET passwd"; content:"|00 01|"; depth:2; content:"pas
   swd"; offset:2; nocase; classtype:successful-admin; sid:2101443; rev:5;)
```

图 6-37　snort 规则

这条规则的报警就可以表示攻击者获得了对根用户和其他用户的控制权。类似这样的例子还有用于检测 BSD Telnet 守护进程攻击的规则。打开 OSSIM 2.3 系统中 Snort 规则库的 telnet.rules 文件，查看第 38 行，如图 6-38 所示。

```
38 alert tcp $EXTERNAL_NET any -> $TELNET_SERVERS 23 (msg:"TELNET bsd exploit client finishing"; fl
   ow:to_client,established; dsize:>200; content:"|FF F6 FF F6 FF FB 08 FF F6|"; depth:50; offset:2
   00; rawbytes; metadata:service telnet; reference:bugtraq,3064; reference:cve,2001-0554; referenc
   e:nessus,10709; classtype:successful-admin; sid:1253; rev:18;)
```

图 6-38　snort 规则

6.6.7　拒绝服务

再举一个“发现 Chargen/echo DOS 攻击”的例子，chargen 服务端口为 UDP 19，echo 为 UDP 7，如果 UDP chargen server 收到一个封包后，就发回一个封包，如果发现与客户端连接存在，会不断发送封包，被利用为 Ddos 攻击后，这样攻击就会被放大，以致占满整个网络带宽。这里攻击者伪造了数据包，将两台开放 Chargen/echo 的服务器互指，如图 6-39 所示，这样结果造成资源耗尽而导致网络不可用。

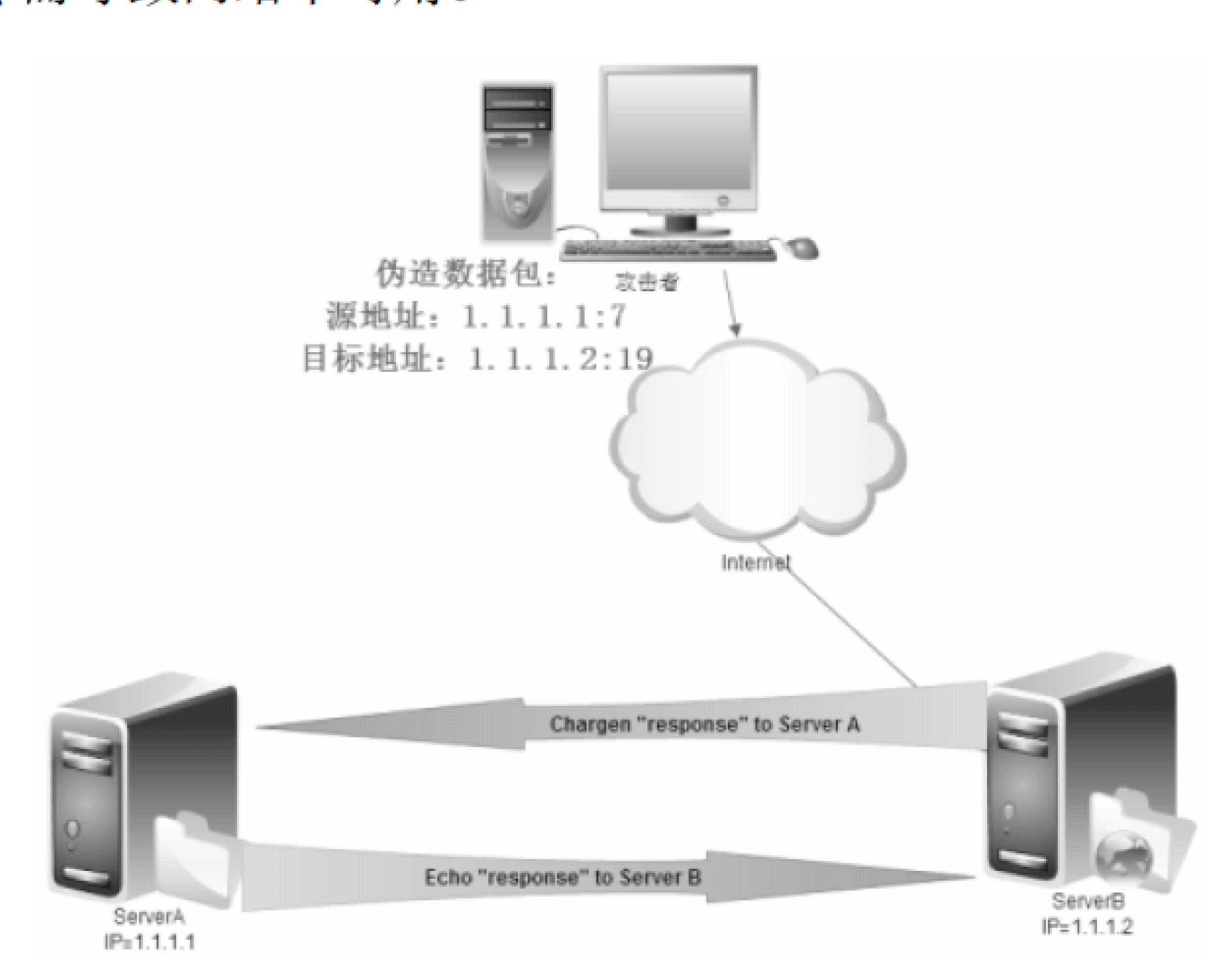


图 6-39　拒绝服务攻击的例子

打开 dos.rules 的第 1 行，如图 6-40 所示。

```
alert udp any 19 <> any 7 (msg:"DOS UDP echo+chargen bomb"; flow:to_server; reference:cve,1999-0103;
 reference:cve,1999-0635; classtype:attempted-dos; sid:271; rev:9;)
```

图 6-40　dos 规则

该特征检测的 DOS 条件是一个 Chargen/echo 服务的无限循环。还有一些 DOS 攻击会出现异常的数据输入，这些输入的内容会使服务器瘫痪，当然，这是个比较老的漏洞。Microsoft FTP STAT globbing DOS 就是一个典型，当攻击者在 Windows 2000 IIS 5.0 的 FTP 服务器上发送 STAT 命令之

后，再追加一些异常的文件名匹配字符（$g.*?等）时，这时没有打补丁的 FTP 服务器将会立即崩溃。我们看看 Snort 应对此种攻击的规则代码，打开 ftp.rules 规则，查看 66 行，如图 6-41 所示。

```
66 alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT STAT * dos attempt"; flow:to_serve
   r,established; content:"STAT"; fast_pattern:only; pcre:"/^STAT\s+[^\n]*\x2a/smi"; metadata:servi
   ce ftp; reference:bugtraq,4482; reference:cve,2002-0073; reference:nessus,10934; reference:url,w
   ww.microsoft.com/technet/security/bulletin/MS02-018.mspx; classtype:attempted-dos; sid:1777; rev
   :13;)
```

图 6-41 snort 规则

DDOS 攻击利用大量的受害主机向目标主机发送大量请求，造成其瘫痪。这种行为也可以用 Snort 规则来检测。以下是在受到 DDOS 攻击时，在 ACID 系统里发出的事件报警信息。

```
#(2 - 10994) [2012-08-21 09:58:06] [arachNIDS/253]  DDOS shaft synflood
IPv4: 195.27.218.62 -> 123.xxx.yyy.252
     hlen=5 TOS=0 dlen=40 ID=28789 flags=0 offset=0 TTL=19 chksum=17659
TCP:  port=13000 -> dport: 13000  flags=******S* seq=674711609
     ack=39679596 off=5 res=0 win=25622 urp=39897 chksum=44744
Payload: none
---------------------------------------------------------------------------
#(2 - 10993) [2012-08-21 09:58:06] [arachNIDS/253]  DDOS shaft synflood
IPv4: 195.27.218.62 -> 123.xxx.yyy.250
     hlen=5 TOS=0 dlen=40 ID=28789 flags=0 offset=0 TTL=19 chksum=17661
TCP:  port=13000 -> dport: 13000  flags=******S* seq=674711609
     ack=1883199726 off=5 res=0 win=20738 urp=46580 chksum=22367
Payload: none
```

以下是 Snort 得到的数据包分析。

```
Snort:
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/21-09:37:16.080331 195.27.218.62:13000 -> 12.82.128.178:13000
TCP TTL:16 TOS:0x0 ID:39977 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0x26917D08  Win: 0x2240  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
Snort processed 1 packets.
Breakdown by protocol:
Action Stats:
    TCP: 1        (100.000%)        ALERTS: 0
    UDP: 0         (0.000%)         LOGGED: 0
   ICMP: 0         (0.000%)         PASSED: 0
    ARP: 0         (0.000%)
   IPv6: 0         (0.000%)
```

原本发起一个序列号为 SYN 包 674711609，将产生一个 SYN/ACK 674711610 或者 ACK 674711610，也许是程序员疏忽，当实际情况是发起一个 SYN 包，它们为每个数据包使用了相同的 TCP 序列号。所以简单地通过检测 674711609 的序列号，就能检测这种攻击，另外，还会发现序列号变成了 0x28374839，为什么?其实十进制的 674711609，表示为十六进制为 0x28374839。接下来我们来看看 Snort 规则如何防范这种攻击，规则比较简单，打开 deleted.rules 的第 105 行，如图 6-42 所示。

```
105 alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg:"DELETED DDOS shaft synflood"; flow:stateless
    ; flags:S,12; seq:674711609; reference:cve,2000-0138; classtype:attempted-dos; sid:241; rev:14;
    )
```

图 6-42 Deleted 部分规则

6.7 高速网络环境的应用

在高速复杂的网络环境下，CPU 处于频繁中断状态，造成接收数据包效率降低。假想一个场景，我们使用标准 100MB 网卡，实际达到的接收速率为 80Mbits/s，当数据包最大长度为 1500Bytes，CPU 每秒产生中断数目为 6,667 个，当数据包最短长度达到 46Bytes，CPU 每秒产生中断数目暴涨到 217,391，是原来的 32 倍。数据包从网卡驱动到内核驱动，再从内核到用户空间数据包被多次复制，浪费了 CPU 的大量时间和资源。在 64 位 OSSIM 中采用 PF_ring 技术，减少了数据包在内存复制次数，彻底解决了这个问题，所以目前 64 位 OSSIM 5 能用在千兆网络环境中。

6.7.1 Suricata VS Snort

多年来 Snort 已成为普遍使用的轻量级开源入侵检测系统，但随着多核及高速带宽时代的来临，单线程的 Snort 已不能满足人们的需要。随着告警事件数量的增加，队列会越来越大，索引深度增加，找到未处理的记录速度随之变慢，直到 Suricata 的出现改变了这一切，表 6-11 的内容简单对比了 Suricata 和 Snort 的主要差异。

表 6-11 Suricata 与 Snort 主要区别

参数	Suricata	Snort
规则	VRT::Snort rules Emerging Threats rules	
线程	多线程	单线程
日志记录	unified2 logs for barnyard、database	
IPv6	支持	默认不支持，编译时需要加入“--enable-ipv6”选项
抓包方式	PF_ring	Libpcap
离线分析	支持	
前端	Sguil、BASE、Snortsnarf	

如果你需要了解当前系统 Suricata 的配置情况可使用如下命令：

```
#suricata  --build-info
```

6.7.2 PF_ring 工作模式

PF_ring 是一种新型的网络 Socket，性能优于 Libpcap，它可以提高包捕获的速度。PF_ring 有三种透明模式（transparent_mode）：

- 0:0 模式：走的是 Linux 标准的 NAPI（New API）包处理流程，我们在 OSSIM 下输入“ethtool –i eth0”，能查看到 NAPI 版本号（目前版本：7.3.21-k5）。

- 1:1 模式：数据包即走 Linux 标准包处理流程，还复制一份给 pf_ring。
- 2:2 模式：驱动只将包复制到 pf_ring，内核则不会接收这些包。

1 和 2 模式需要特殊网卡驱动的支持，一般 PF_ring 为 0 模式。/etc/modprobe.d/pfring.conf 这个配置文件中设置了工作模式为 0，pf_ring 模块位于 /lib/modules/2.6.32-5-amd64/kernel/net/pf_ring/目录。

6.8 网络异常行为分析

普通的流量监控软件很难感知到网络异常行为，这也是运维人员在监控领域的一块短板。所谓异常行为主要指通过计算机和互联网，对网络中的各种设备上的原始数据进行恶意的操作（增加、删除、查询和修改），从而达到破坏操作系统功能，或者对用户有危害的行为活动。例如，目前最为常见的网络异常行为，主要包括木马病毒、黑客攻击、恶意软件的一系列行为等。这些异常行为都会对用户的网上活动造成一定的影响，甚至导致网络瘫痪。

6.8.1 流程分析

行为特征提取模块流程：首先是链路层协议分析，并由链路层向高层递进，如果提取到各层协议的特征就结束提取。下面举一个应用层 HTTP 协议跨站脚本攻击的例子，来分析这个模块的流程，如图 6-43 所示。

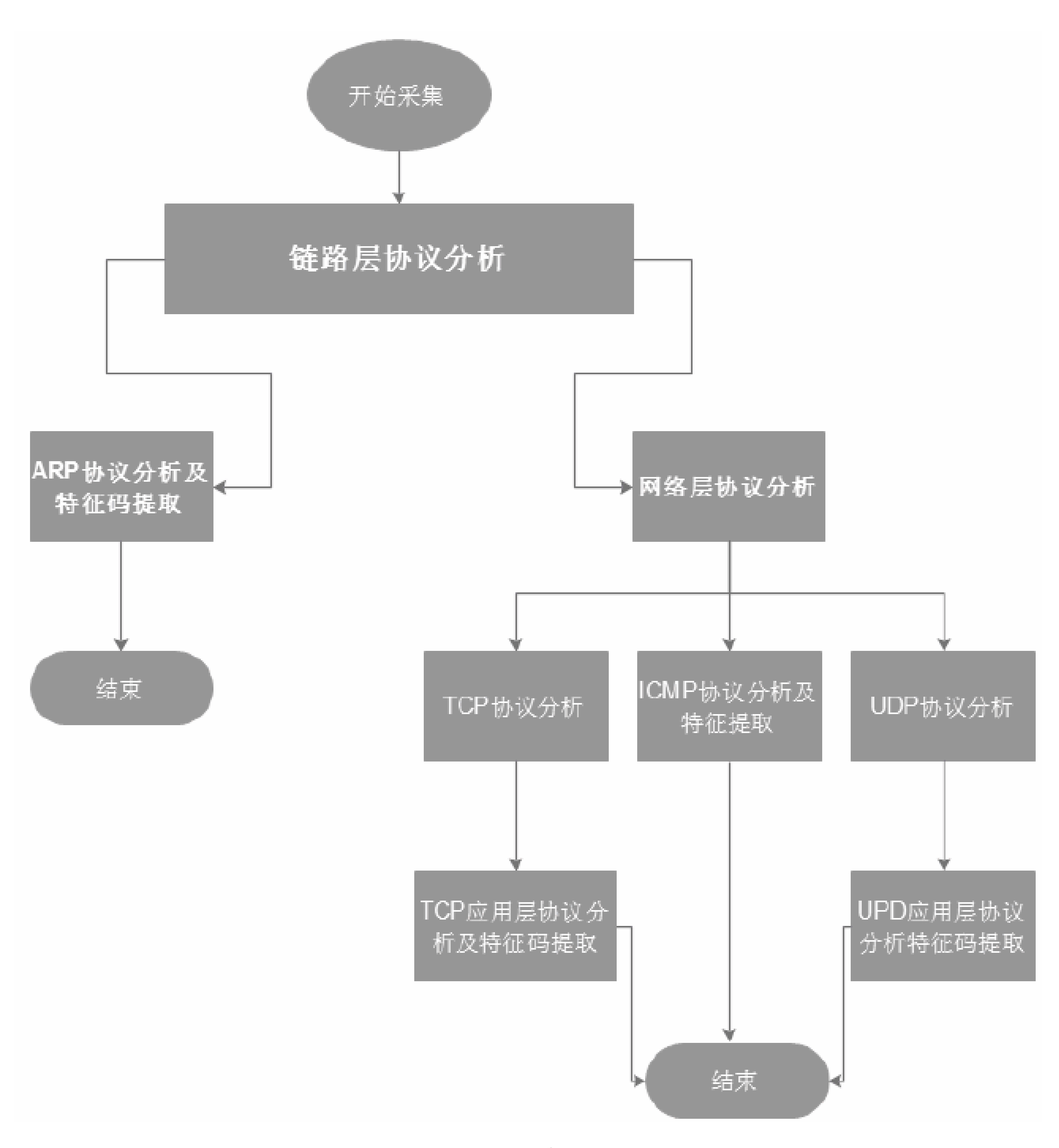


图 6-43　行为特征提取流程

OSSIM 系统通过 Sensor 中的嗅探模式，来实现异常行为分析，实质是通过对网络数据包进行分析，同时匹配网络行为特征库实现。行为匹配模块又包括字符串匹配、协议匹配、长度匹配和逻辑匹配。

（1）字符串匹配是系统最主要的匹配方式之一。根据对某个网络异常行为数据包的深度分析，提取出数据包内容字段中字符串特征。通常该系统经过协议分析后，再进行字符串匹配。此外，还有数据包大小匹配，它也属于字符串匹配。这种匹配方式只针对数据包某段内容字段中的长度匹配，而不是具体的字符串进行匹配。

（2）数量匹配是通过对某些事件出现的数量来产生新的事件。

（3）协议匹配主要通过数据包采集模块，将网络数据包按照协议分析的结果，对协议相应的字段进行检测。协议匹配需要对 TCP/IP 各层的协议进行分析，对应用层协议分析，如 Ntop 服务，可以显著地提高匹配的效率。

为了提高匹配的效率，匹配时按照网络行为库与数据包内容匹配的要求去匹配相应的部分，并不需要对整个包体部分进行匹配。

6.8.2　举例

网络行为库记录了常见网络行为（主要是网络异常行为）的“特征码”，利用它们来与获取到的网络数据包进行匹配分析，以发现异常的网络数据包。

在网络中传输的数据包都有它的内容，也称数据包净荷，一般以十六进制数表示。一条网络行为需要很多属性来描述，包括数据包包头信息、数据包内容、行为描述等。数据包包头包括 IP 头、TCP 头、UDP 头、ICMP 头等，其中 IP 头记录了数据包的源地址、源端口、目标地址、目的端口以及其他一些 TCP/IP 中规定的 IP 头应有的字段。

Exploit/shellcode

例如 Shellcode 代码：

```
#include "stdio.h"
#include "stdlib.h"
char shellcode[] =
"\x33\xc9\x83\xe9\xdd\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x98"
"\x44\x82\x1c\x83\xeb\xfc\xe2\xf4\x64\xac\xc6\x1c\x98\x44\x09\x59"
"\xa4\xcf\xfe\x19\xe0\x45\x6d\x97\xd7\x5c\x09\x43\xb8\x45\x69\x55"
"\x13\x70\x09\x1d\x76\x75\x42\x85\x34\xc0\x42\x68\x9f\x85\x48\x11"
"\x99\x86\x69\xe8\xa3\x10\xa6\x18\xed\xa1\x09\x43\xbc\x45\x69\x7a"
"\x13\x48\xc9\x97\xc7\x58\x83\xf7\x13\x58\x09\x1d\x73\xcd\xde\x38"
"\x9c\x87\xb3\xdc\xfc\xcf\xc2\x2c\x1d\x84\xfa\x10\x13\x04\x8e\x97"
"\xe8\x58\x2f\x97\xf0\x4c\x69\x15\x13\xc4\x32\x1c\x98\x44\x09\x74"
"\xa4\x1b\xb3\xea\xf8\x12\x0b\xe4\x1b\x84\xf9\x4c\xf0\xb4\x08\x18"
"\xc7\x2c\x1a\xe2\x12\x4a\xd5\xe3\x7f\x27\xe3\x70\xfb\x6a\xe7\x64"
"\xfd\x44\x82\x1c";
int main (int argc,char *argv[])
{
int *ret;
ret=(int *)&ret+2;
*ret=(int)shellcode;
}
```

此时会触发 SIEM 报警，如图 6-44 所示。

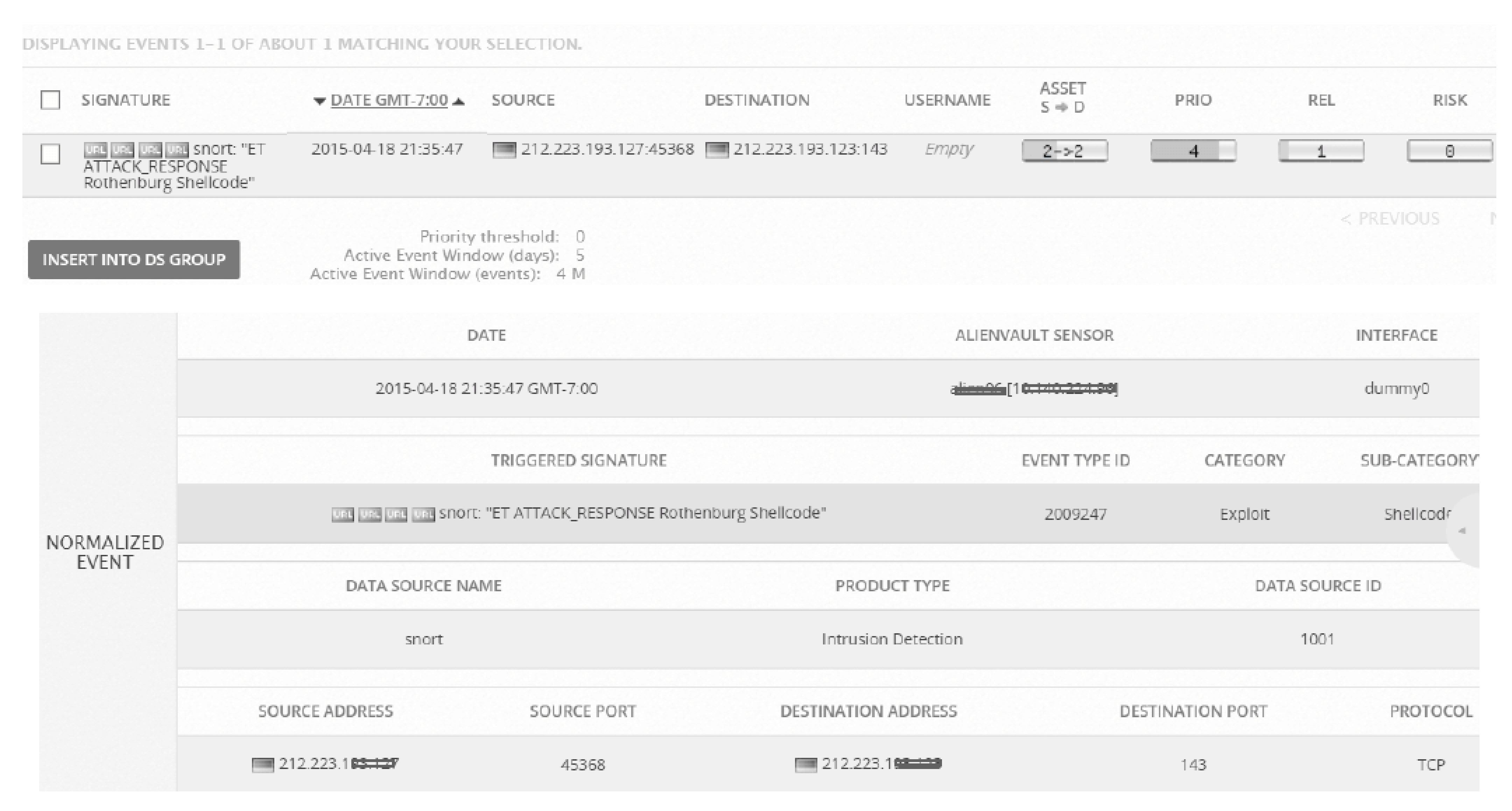


图 6-44　典型 SIEM 报警

网络异常行为都在产生新的变异，因此网络行为库必须时常更新，添加新的行为规则。一般情况下，有两种更新网络行为库的方法（该方法对网络行为提取人提出了很高的要求，需要精通网络知识且具备丰富的实践经验），一种是通过实验蜜罐系统从 Internet 中提取网络行为以更新网络行为库，另一种是通过 Snort 系统的规则库提取网络行为以更新网络行为库。

6.10 小结

本章重点讲解了 Snort 预处理器，snort 日志分析方法和可疑流量和异常行为分析方法，包括 snort 规则分析和编写。OSSIM 中 Snort 内置的一些规则，只用来学习还不够，要想发挥作用，还需要结合企业自身的业务，在网络风险和威胁不断变化中，经常验证这些规则是否有效，及时更新和修改规则。

第三篇

实 战 篇

Open Source Security
Information Management
Big Data
Cloud Computing
Internet of Things

第 7 章

OSSIM日志收集与分析

从本章可以学习到：

- Syslog 协议
- Rsyslog 配置方法
- 网络设备日志分析
- Linux 各种服务的日志分析
- OSSIM 日志管理
- 用 Snare/WMI 收集 Windows 日志

以铜为镜，可以正衣冠，以人为镜，可以明得失。我们分析日志就能了解系统中的问题。通过本章的内容讲解掌握如何获取并分析各类系统的日志，学习好本章内容将为后续章节的案例分析打下坚实的基础。

当企业网络的规模及应用系统不断增大时，如何确保网络运营环境的稳定、安全、高效呢？我们先看看系统管理员不得不面临的问题——如何监控用户的网络应用行为?如何跟踪网络应用资源的使用情况?如何识别网络中的异常流量和性能瓶颈?如何有效地规划和部署网络资源?如何迅速响应网络的故障告警？日志技术可以记录系统产生的所有行为，并按照某种规范表达出来。日志系统指收集网络设备的事件信息的一套机制。我们可以使用日志记录的信息为系统进行排错，并优化操作系统性能。

7.1 日志分析现状

企业信息系统中会包含多种设备，如路由器、防火墙、IDS/IPS 、交换机、服务器和 SQL 数据库等。各种复杂的应用系统和网络设备每天都会产生大量的日志，如果不加以收集，一旦出了问题需要查找如此海量的数据，对于管理员来说如同是一场噩梦。

目前，不少企业网络环境中，日志的存储和分析没有受到重视，往往在平时忙碌的工作中被忽视掉，只有在系统提示磁盘空间不足或者系统瘫痪了才引起管理员注意。例如，在 Windows 服务器中频繁出现报警，这时从 Windows 事件查看器的应用程序日志里看到内容为：“Windows（2592）Windows: 由于系统错误 112（0x00000070）:磁盘空间不足。”类似上述情况同样会发生在 Linux/Unix 系统中，如果处理不及时，很可能导致系统宕机。例如在一台

Linux 服务器上检查到一段日志信息：

```
    May 20 01:13:00  www.write.com kernel: pdi 1503 (dd),udi 3 inumber 9833 on /var:
filesystem full
```

从上面这段日志就能发现系统的/var/文件系统已满，继续检查发现/var/spool/、/var/squid/cache、/var/www/、/var/lib/mysql/这几个目录占用磁盘空间很大，需要用合适的方法处理。

另一种情况，当磁盘空间不足时，一些管理员首先想到简单地移走文件或直接删除，只考虑日志会占用空间，没考虑到日志今后可以作为司法取证的分析材料，因为在日志里可以发现故障发生的时间等关键信息。

7.1.1 日志记录内容

不同设备记录日志格式不同，那么什么样的日志才是有利于审计和有利于安全信息平台使用呢？目前国际上还没有统一的标准去衡量，但是通常来讲完整的日志应该包含如下信息：

- Who：谁登录了。
- What：他们在做什么，又发生了什么。
- Where：发生在何处，在哪台主机，哪个文件系统上，或者哪个网络接口等等。
- When：何时发生，应该包含开始时间和结束时间。

为了得到更详细的内容，有时候还希望得到：为什么会发生（Why），如何发生（How），以及发生发展的趋势是什么，简单说就是 5W1H。如图 7-1 所示。

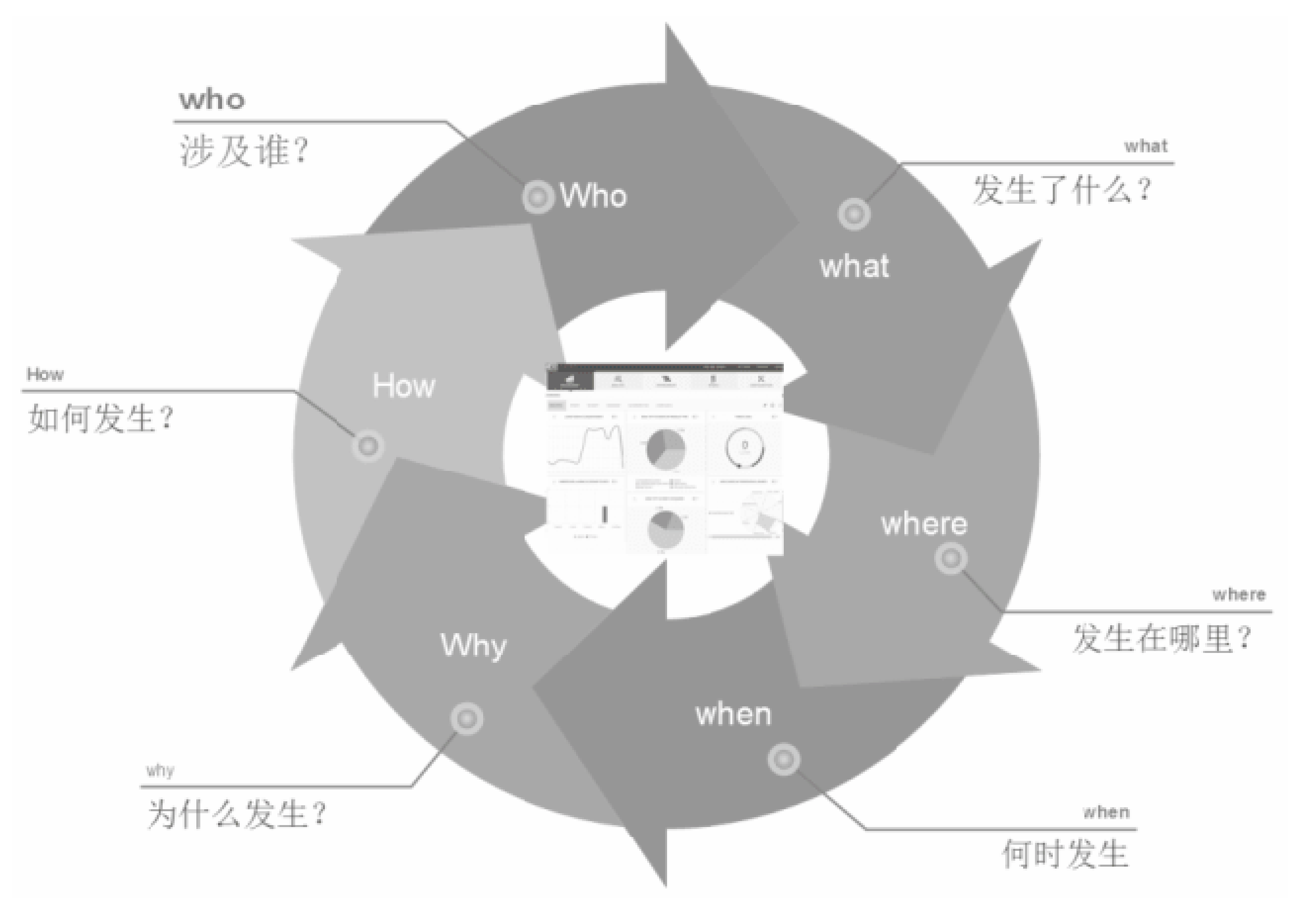


图 7-1　5W- 1H

有关安全运营的日志记录主要包含以下几个方面：

（1）身份认证和授权。

（2）系统变更管理（记录系统变更或组件变更等）。

（3）网络威胁管理（由防火墙，IPS 设备产生）。

（4）性能和容量管理（包括系统各种阈值，如 CPU 利用率、内存、网络带宽占用等）。

（5）业务的可持续性管理（系统开关机和冗余备份相关功能）。

7.1.2　日志中能看出什么

数据是企业的财富，同样日志蕴含着丰富的信息。Ping 命令是在平常不过的网管工具了，使用 ICMP 来 ping 对方主机，就知道是否存活。但在故障时，能够 ping 通对方的网络接口，不代表机器正常，也有可能系统已出现故障，这样就需要从日志中获取详细信息。日志不仅告诉你主机是否存活，还能够告诉你主机上运行应用程序的状态，以及它们在做什么，在真正宕机前发出各种日志报警信息。以下列举了某主机 SSH 攻击日志，我们看看能读出那些有用的信息：

```
    Sep 17 07：00：02 webserver.com:sshd[20392]:Failed password for illegal user test
from 192.168.11.2 port 33783 ssh2
```

从该日志可以看出，用户 test 登录失败，而且他是非法用户，这条消息表明，一个攻击者使用扫描器探测服务器。试图利用字典里的用户进行试探，但攻击还没成功。

当然，一旦攻击成功，黑客接管主机后完全有可能删除这条日志（为了避免日志被删除，可以将这条日志发送到远程的日志收集服务器集中存储）。接着看下面的日志。

```
    Sep 17 10:09:10 webserver.com adduser[2341]: new user:name=test, uid=0,gid=0,
home=/root/cgi,shell=/bin/bash
```

很显然，从这条消息看出，系统添加了系统管理员账户名称为 test，用户 ID 为 0（相当于 root 权限用户）。

用户行为会被日志记录下来，比如用户登录、注销和启动进程等，系统审计工具能提供细颗粒的日志记录，而且这种日志一旦被系统记录并收集之后，就不会因系统故障恢复正常而被修改，也就是说日志记录了系统从正常→故障→正常的完整过程，由于每条日志都有时间戳，它们提供了每个事件的时间顺序，日志不仅告诉我们发生了什么，还告诉我们事件发生的时间和顺序，而且日志发送到日志服务器后也提供了独立的日志收集仓库，一旦原始主机上的日志遭到破坏（比如篡改和删除），独立的日志搜集服务就是可靠的日志附加来源。

例如，在 Linux 中普通用户通过 sudo 来执行管理员命令，无意间删除了重要的系统文件而导致故障，在取证时，sudo 日志提供了取证日志。又如，用户张三在很短的时间内，从地域跨度很大的两个不同的位置，用 SSH 方式登录系统，这种异常行为就非常可疑。

7.1.3　日志分析的基本工具及缺陷

在 Unix/Linux 中内置了强大的命令行日志分析工具，例如 grep、awk、tail、sed 等，用它们分析简单的日志没有问题 然而随着日志尺寸的不断增大 这种分析工具也暴露出一些问题：

（1）不同日志之间无法关联，用人工分析法无法同时分析来自不同网络设备的日志，更无法搞清楚整个网络攻击是如何发生发展的。

（2）在超过一 GB 大小的日志中检索信息非常困难。

（3）Grep 这样的工具只能对明确的关键字信息检索，无法生成统计趋势和异常行为。

以上问题都可以在 OSSIM 系统的日志分析平台中得到解决。

7.1.4 海量日志收集方式

一个海量日志收集系统，首先要考虑采用什么样的日志采集模型。一般有两种方式，推送（PUSH）方式和拉（PULL）方式。我们常见的 Syslog、Rsyslog 就是采用推送 (PUSH)方式，再比如 Facebook、Scribe 也采用了 PUSH 方式，这种方式要求日志收集存储容量大，至少要大于峰值时的数据生成量，否则主动推送过来的数据将来不及处理，另外如果设置不正确，会收到虚假的 syslog 信息，所以必须了解那些设备启用了 syslog。而拉（PULL）方式的日志收集并不常用。

7.2 日志消息格式与存储

7.2.1 日志消息格式

以 OSSIM USM 企业版为例，讲解通过 rsyslog 收集网络设备的日志，下面看一个 Cisco 交换机传来的日志，如图 7-2 所示。

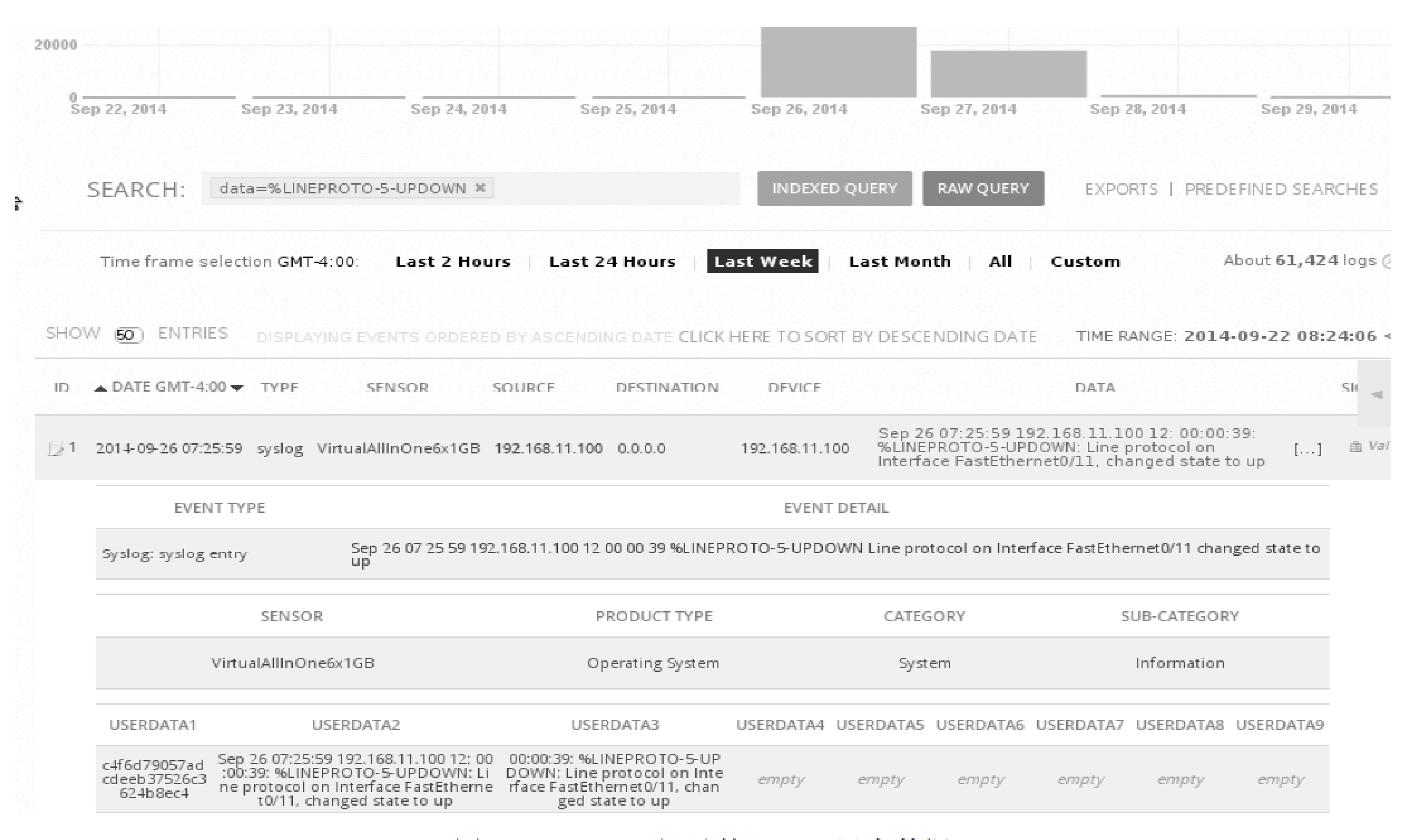


图 7-2 OSSIM 记录的 syslog 日志数据

在图 7-2 中由 Cisco 交换机生成的 Syslog 消息，其中 DATA 栏的信息包括日志消息的时间戳，内容有接收的月日时分秒。192.168.11.100 是交换机的 IP 地址， 后面冒号的作用是和日

志消息内容分开。在%LINEPROTO-5-UPDOWN 中，数字 5 表示优先级，5 级表示严重事件。

7.2.2　OSSIM 下的日志查询比较

在 OSSIM USM 中，输入关键字系统会自动匹配数据源，如图 7-3 所示。

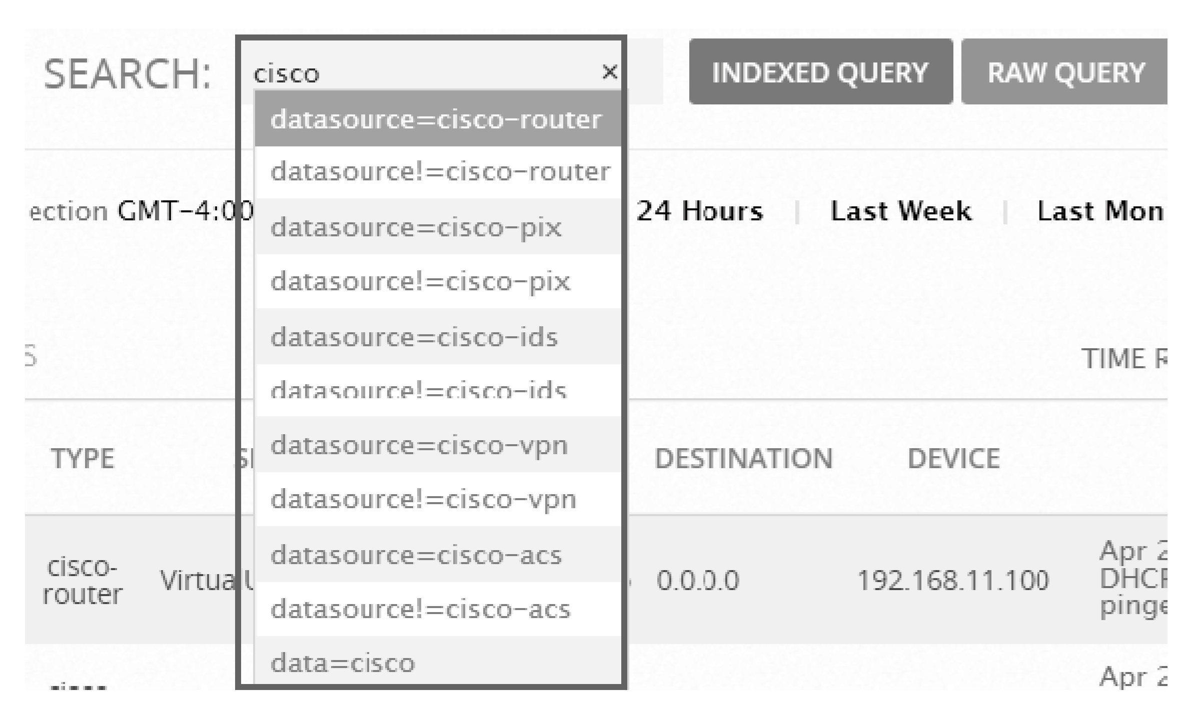


图 7-3　自动匹配关键词

查询日志分为索引查询（Indexed Query）和 RAW（Raw Query）查询两种，索引查询是最快速的方法，而 RAW 日志查询主要是查询存储在/var/ossim/logs 目录下的文本文件。RAW 查询会通过正则表达式直接匹配你所检索的关键字，会直接读取这些文本文件。下面的实验通过查询 Cisco-Router 日志来对比这两种方法的查询速度。

首先选择 RAW Query 查询，查询范围选择 All，表示对所有日志进行查询，如图 7-4 所示。

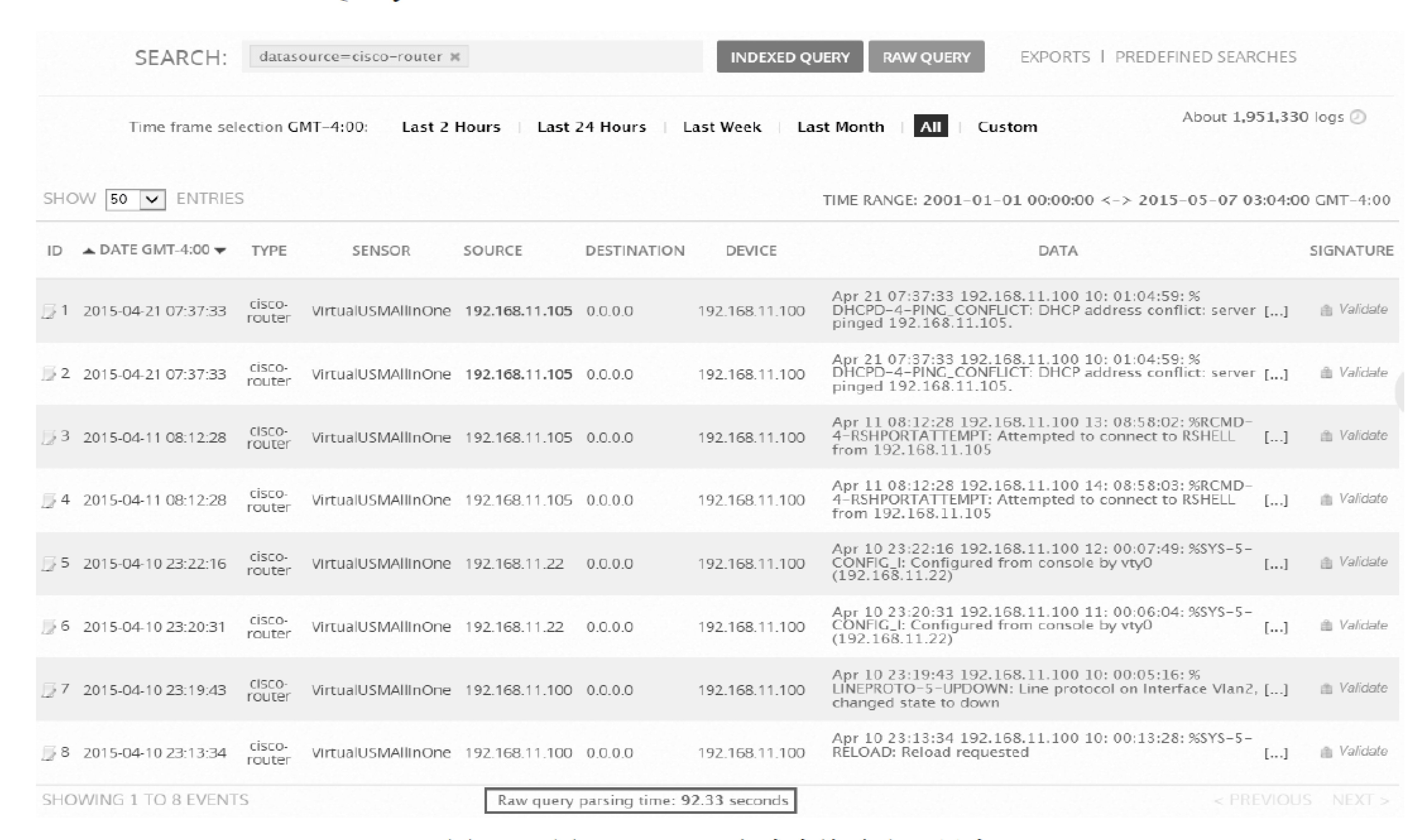


图 7-4　用 Raw Query 方式查询路由器日志

在 195 万条日志中查询，花费的时间为 92s。接下来还是同样的查询范围，选择“Indexed Query”，如图 7-5 所示。

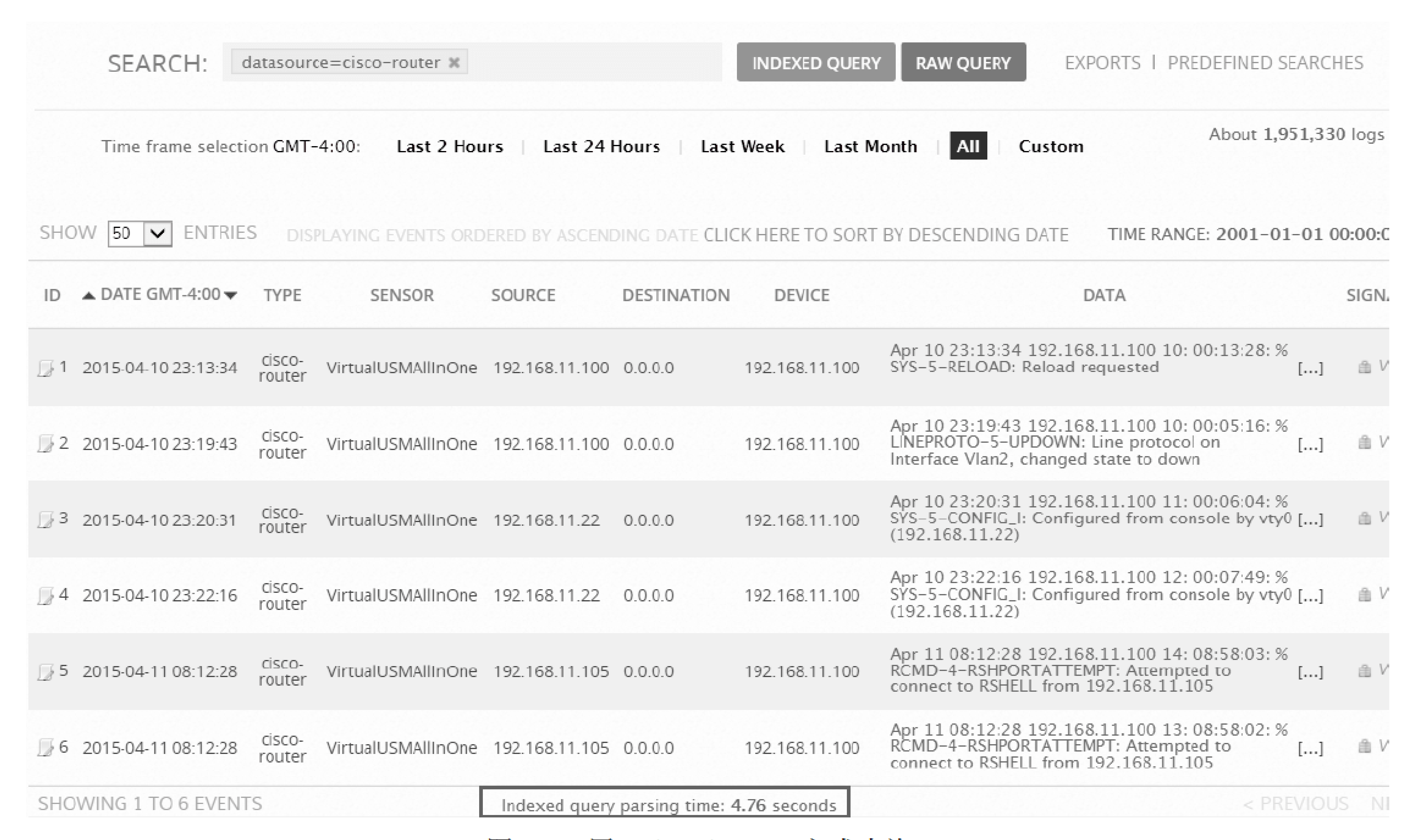


图 7-5　用 Indexed Query 方式查询

这次查询时间仅为 4.76s。为了缩短查询时间，大家可以采用自定义时间区间的方法，可减小查询负担。在 Custom 按钮的右侧，可以查看最近一次系统建立索引的时间，如图 7-6 所示。

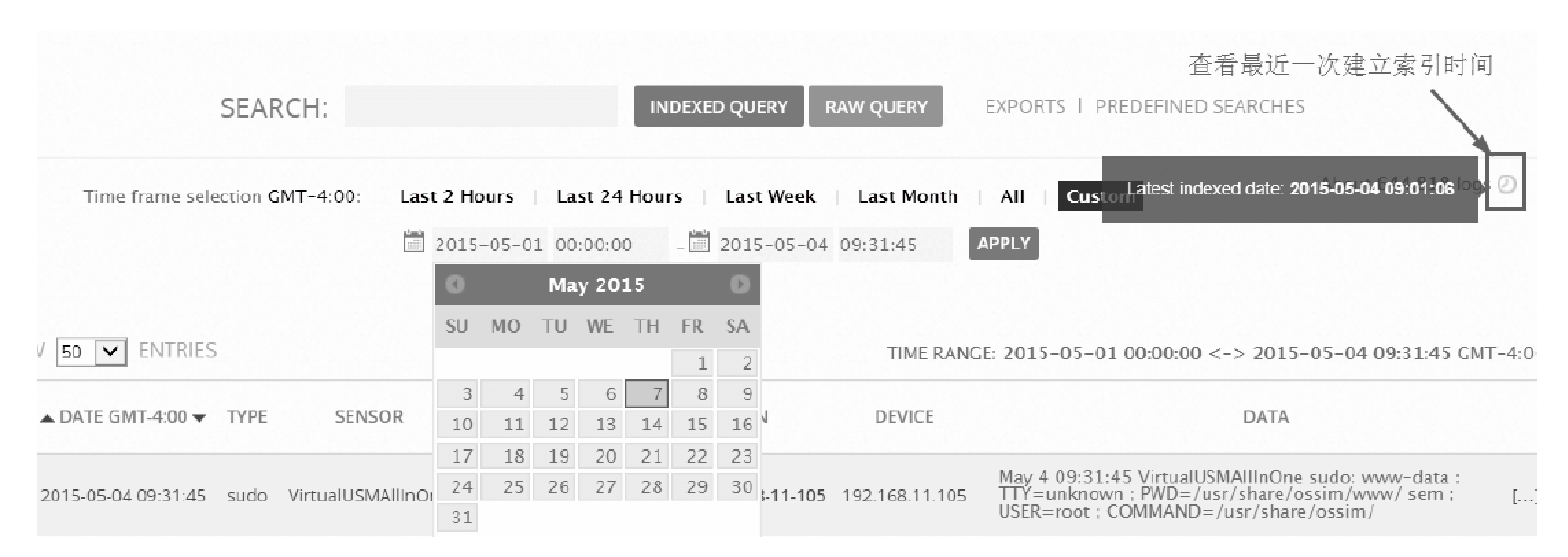


图 7-6　根据日期查询

7.2.3　日志的导出

我们可以选择“Exports”按钮，将查询结果导出，当你输入查询条件和时间范围之后，

单击“Screen export”，系统生成一条记录，如图 7-7 所示，单击按钮就可以直接下载。

SEARCH: datasource=ssh
输入查询条件
Time frame selection GMT-4:00: Last 2 Hou
INDEXED QUERY　RAW QUERY　EXPORTS | PREDEFINED SEARCHES
SAVED EXPORTS
Screen export　Entire export　选定时间范围

FROM	TO	QUERY	SIZE
2015-05-04 11:00:00	2015-05-04 13:57:45	yes	5KB
2015-05-04 11:00:00	2015-05-04 13:52:06	yes	6KB
2015-05-01 04:00:00	2015-05-04 13:31:45	yes	10MB

YPE　SENSOR　SOURCE　DATA　: 2015-05-04 07:00:00 <->

图 7-7　导出某段时间范围内的 SSH 日志

下载的文件是 ZIP 格式，例如 admin_20150504110000　20150504135206_none_d41d8cd98f00b204e9800998ecf8427e.zip，首先需要解压缩。内容如图 7-8 所示。

```
C:\ossim>more loglist.txt
/var/ossim/logs/2015/05/04/13/192.168.11.105/2015-05-04T13-00-00.015095Z.log

C:\ossim>more results.txt|more
May  4 09:52:06 VirtualUSMAllInOne sudo: www-data : TTY=unknown ; PWD=/usr/share
/ossim/www/sem ; USER=root ; COMMAND=/usr/share/ossim/www/sem/test_remote_ssh.pl
 127.0.0.1
May  4 09:52:01 VirtualUSMAllInOne /USR/SBIN/CRON[10320]: (root) CMD ( /usr/shar
e/ossim/scripts/vulnmeter/nessus_jobs.pl -c >> /var/log/ossim/nessus_jobs 2>&1)
May  4 09:51:59 VirtualUSMAllInOne nfcapd[13501]: Ident: 564DB4303295CB66AE8A814
1C00F6233,  Data length error: too little data for common netflow header. cnt: 1
```

图 7-8　查看收集到的日志

通过预定义查询（Predefined Searches）可以将经常查询的关键字存储起来方便下次查阅。

7.2.4　日志分类可视化

OSSIM USM 中可按年月日的方式显示收集来的日志，如图 7-9 所示，我们可以单击图中右侧所示的扇形图标，它的功能是将日志分类显示。

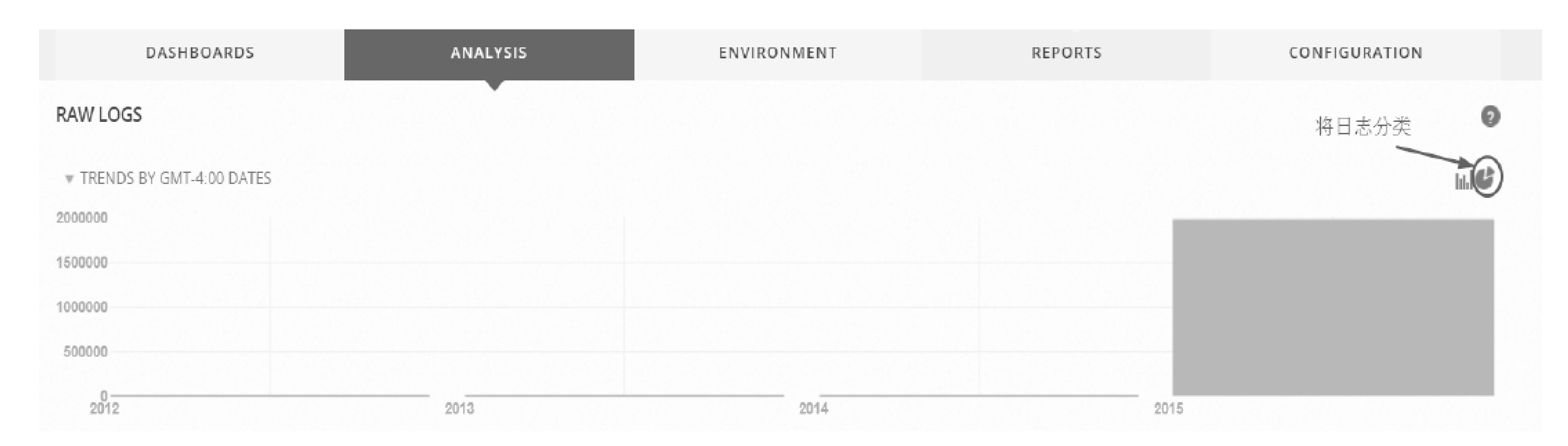


图 7-9　日志显示

当单击图 7-9 所示的日志分类图标后，系统自动将日志根据数据源进行分类，还能根据传感器、事件类型、源和目标地址分类显示，分析涵盖了对事件的归类统计及事件的变化发展趋势，如图 7-10 所示。

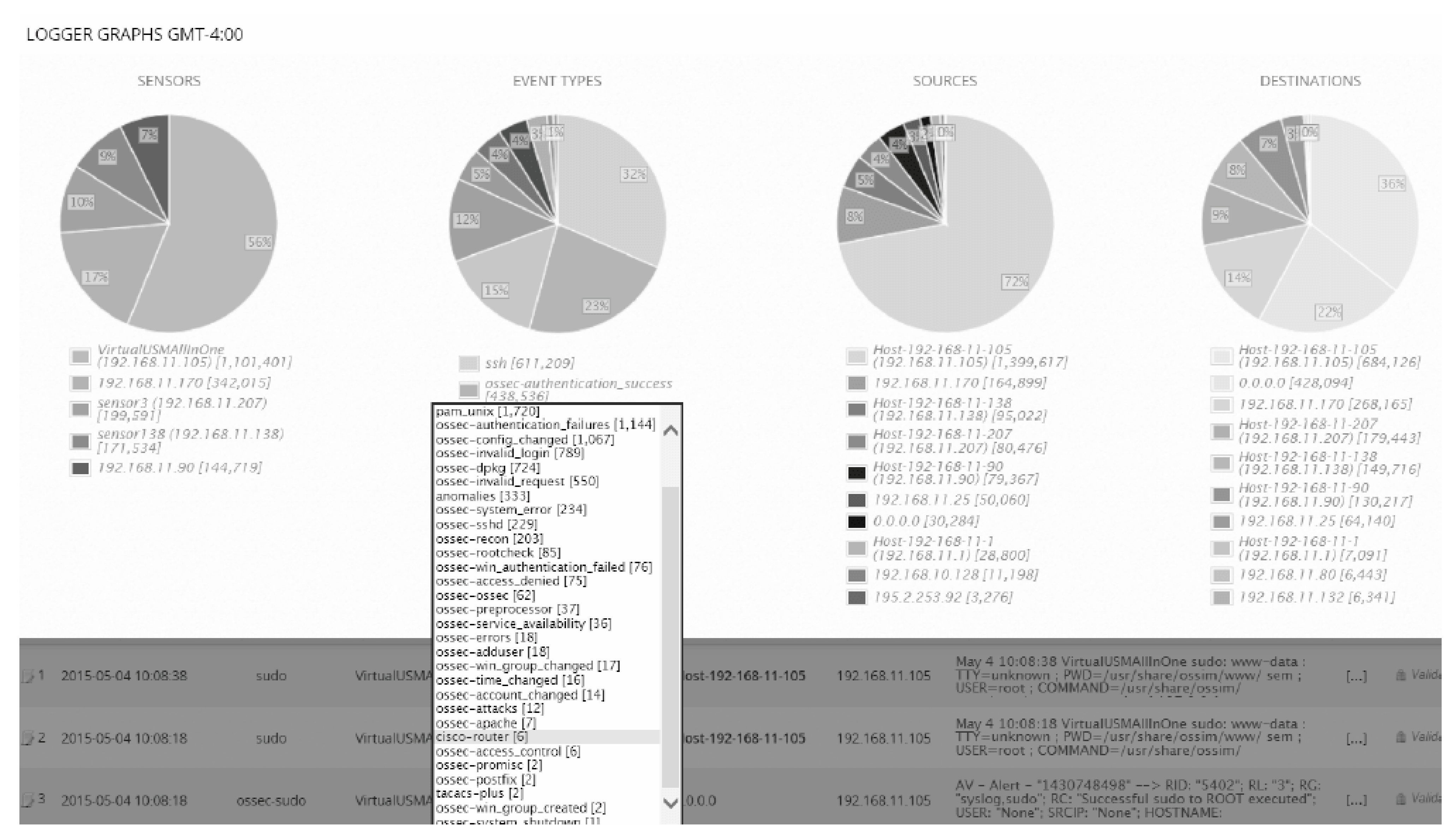


图 7-10　日志按数据源分类显示

复杂的日志经过数据处理之后，摇身一变成了直观的图形化日志，便于管理，还有更多可视化日志分析小工具 Gltail、Logstalgia 及 Gource 参见博文 http://chenguang.blog.51cto.com/350944/1329318。

7.2.5　基于文本格式的日志

由于应用程序生成基于文本的日志对系统开销（CPU 以及磁盘 I/O）而言比较低，阅读性比较强，能够使用 grep、more、awk、sort 查询及过滤处理，所以大量出现在 UNIX/Linux 系统中。

系统通常首先创建一个空文件，例如/var/log/apache.log，向其中不断写入新的日志条目，直到大小超过一定限制，然后再分成另一个文件，例如 access.log.1、access.log.2、access.log.3（我们可以在/etc/logrotate.conf 查看配置）并继续写入，直到分区/var 空间耗尽为止。又如在 OSSIM 系统下的 OSSEC 日志，如下所示。

```
VirtualAllInOne6x1GB:/var/ossec/logs/alerts/2014/Sep# ls -l
total 18196
-rwxrw---- 1 www-data ossec 14143348 Sep 26 19:45 ossec-alerts-26.log
-rwxrw---- 1 www-data ossec  4388089 Sep 27 23:57 ossec-alerts-27.log
-rw-r----- 1 ossec    ossec     2056 Sep 29 00:02 ossec-alerts-28.log.gz
-rw-r----- 1 ossec    ossec      334 Sep 29 00:02 ossec-alerts-28.log.sum
-rwxrw---- 1 www-data ossec    52918 Sep 29 12:31 ossec-alerts-29.log
```

从上面看出，日志根据接收事件的年、月来组织，所以要想检索 2014 年 9 月之后的日志，我们只要进入/var/ossec/logs/alerts/2014/Sep 目录下查询即可，而不用查询所有 ossec 保存日志。从这些日志生成时间还能看出，它们是根据时间先后顺序排序。我们把这种文件称为扁平文件，之所以这么取名是因为这样的文件是扁平的、无模式的一种自由格式，查询和阅读都非常

方便。无独有偶，在 MS IIS 服务器的日志格式为 ex+年份的末两位数字+月份+日期，其日志默认位置在%systemroot%\system32\logfiles\下，我们使用任何文本编辑器都能查看。

OSSIM 的日志存储方式和 Ossec 非常类似，也是采用扁平日志+索引的模式，我们在 OSSIM 系统中，/var/ossim/logs/2014/09/26 目录下就能看到当天以小时为单位的日志，为了从宏观上观察这些目录，我们使用了磁盘分析工具展现/var/ossim/logs 目录中数据分布情况，图 7-11 中最外围圆圈的绿色扇形区域表示了每小时所记录的日志，这些日志使用任何编辑器打开。

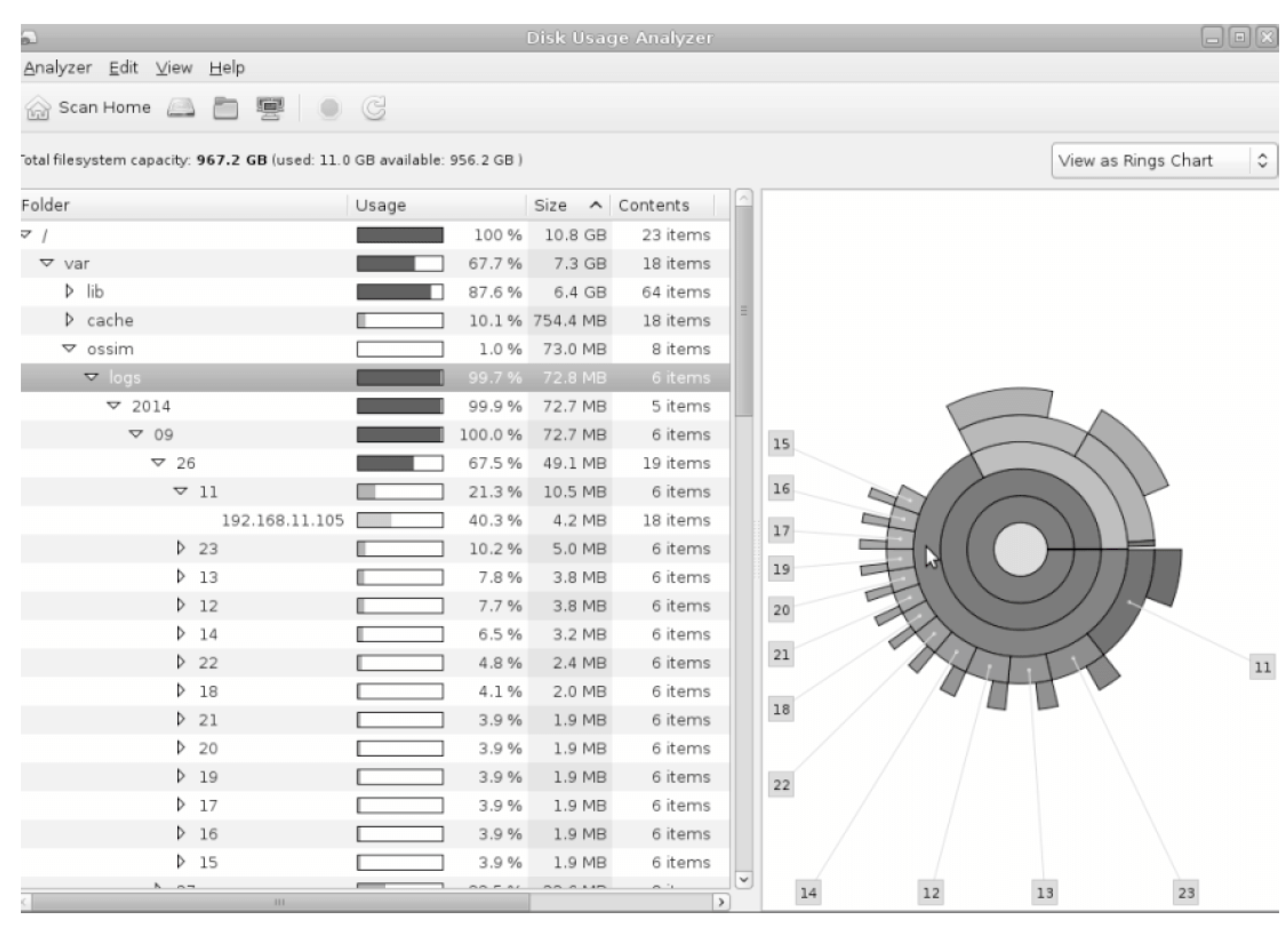


图 7-11　存储日志分布

当然，这种文件在几百 KB 或几 MB 大小时，查询都没什么问题，当文件容量达到几百 MB 甚至是上 GB 大小时，其致命的缺陷也就开始暴露出来，比如我们用 grep 和 sort 组成的 shell 过滤一个含有 10 万条日志的 access.log 文件，往往需要 1 分钟左右时间，这个时间是管理员不能忍受的，这需要按匹配规则逐个字节去匹配，这一过程同时将不符合要求的那些文件逐行地读了一遍，时间全花费在读哪些根本不需要处理的文件上。假设在某目录下有 100 个包含 10 万行的 log 文件，需要在如此大的范围找出某个关键字，那么搜索时间很难想象。解决方法之一就是对上面的扁平日志文件进行索引，使得通过日志的关键元素能够快速查询，也便于对过期的日志进行批量销毁。

7.2.6 基于压缩模式的日志文件

在 UNIX/Linux 中可以使用 tar 将日志压缩，也可以用 zcat 或 zgrep 从压缩文件中读取日志，但这都需要我们手工输入脚本实现，对于一些普通用户显得有些力不从心，logrotate 可以方便地实现日志的轮询、压缩功能，可以在日志增长到一定大小时，或在每天、每周时间到达时，产生新的日志文件。

```
agent.log                  monit.log              reputation.log
agent.log.1.gz             monit.log.1.gz         reputation.log.1.gz
agent.log.2.gz             monit.log.2.gz         reputation.log.2.gz
agent.log.3.gz             monit.log.3.gz         reputation.log.3.gz
agent.log.4.gz             monit.log.4.gz         reputation.log.4.gz
agent.log.5.gz             monit.log.5.gz         reputation.log.5.gz
agent_error.log            nessus_cron.log        rrd_plugin.log
agent_error.log.1.gz       nessus_cron.log.1.gz   rrd_plugin.log.1.gz
agent_error.log.2.gz       nessus_cron.log.2.gz   rrd_plugin.log.2.gz
agent_error.log.3.gz       nessus_cron.log.3.gz   rrd_plugin.log.3.gz
agent_error.log.4.gz       nessus_cron.log.4.gz   rrd_plugin.log.4.gz
agent_error.log.5.gz       nessus_cron.log.5.gz   rrd_plugin.log.5.gz
arpwatch-eth0.log          nessus_jobs            sem.log
```

对于这些日志的处理需要掌握以下方法：

```
#gzip  access.log                              \\压缩文件并生成 access.log.gz 文件
#gunzip access.log.gz                          \\解压缩
#gunzip -c access.log.gz |grep  xx             \\查找.gz 包中的内容
#zcat access.log.gz                            \\查看压缩文件内容
#zmore access.log.gz                           \\查看压缩文件内容
#zdiff  access.log.1.gz  access.log.2.gz       \\比较两个压缩文件差异
#zcat access.log.*.gz  |goaccess               \\批量分析日志
```

在 gzip 压缩文件中搜索，zgrep 与 zcat 很相似，可以用于 gzip 压缩过的文件。它与 grep 有相似的命令选项，例如：

```
#zgrep -i error /var/log/syslog.1.gz
将日志*.log.gz 结尾的文件合并
#cat *.log.gz > file
#zcat $(ls -t *.log.gz) | gzip > new_file.gz
```

7.2.7 日志转储到数据库

很多情况下我们需要将日志存储到数据库，因为那样在创建日志分析报表，以及多台网络设备关联日志分析时非常方便，而且日志信息可以通过标准的 SQL 查询快速搜索到日志记录，而且也便于集中权限管理。从事过数据库管理的读者一定知道，海量日志存入数据库系统中，不但数据量太庞大，而且无法避免地造成数据库系统开销显著增加，因为向数据库中写入同样一个数据，比写入磁盘文本文件所花销的时间要长（包括网络延迟、SQL 解析、索引更新、磁盘写入等时间）。即使在数据库中构建了索引和查询优化，但在上亿行数据中搜索还是会变得非常迟缓。

面对这一问题应该清楚，我们需要具体将哪些日志存储到数据库。作为计算机取证，又要求需要完整的 RAW LOG，也就是未经任何修改的日志。对于 OSSIM 系统而言，既要兼顾性能，又要做到日志完整，所以采取的混合和存储的方式会更好。在 OSSIM 系统中，原始日志通过 Rsyslog 收集存储在 OSSIM Server，同时经过 Agent 归一化处理，并经过提炼的带有一定级别关键事故的日志会被存储在 MySQL 数据库，以便输出报表。

7.2.8　日志处理及保存时间

日志中保存了系统故障、配置错误、数据访问日志、攻击日志等丰富内容，对于归档日志需要进行完整性监控，而且在审计中要求日志留存历史至少一年，保持在线三个月可查询。可参考 2.16.11 节中有关事件保存时间的要求，同时，作为企业的安全人员每天应该审核日志系统的所有组件并查看各种分析报表。

7.2.9　日志系统保护

对于日志的安全，其受重视程度不亚于部队对军火库的重视。攻击者有意无意地会针对日志系统展开攻击，因此我们应对日志集中存储的系统进行更加严格的保护。首先保证日志传输安全（通过 SSL，SSH 或 IPSEC 加密），其次保证日志主机可用，保证关联引擎分析的可用性，防止有时候大量泛洪攻击生成了很多干扰报警使关联引擎处理不过来。

7.2.10　日志轮询

日志轮询相当于日志的多份复制，目的是节省磁盘空间，用于短期分析和存储，它在 OSSIM 中使用的技术是由 logrotate 工具实现。日志轮询可以采用两种策略：

- 基于大小的轮询：日志文件达到某个预设值时轮询，例如 50MB 等。
- 基于时间的轮询：日志文件根据某个设定周期开始轮询，例如每天、每周等。

OSSIM 系统下日志轮询由 logrotate 负责，它的参数配置文件放在两个地方：

- /etc/logrotate.conf，主要的参数文件。
- /etc/logrotate.d/，此目录下的所有文件会被读入 logrotate.conf，如没有指定则以 logrotate.conf 中的为准。

/etc/logrotate.d/包含 apache2、alienvault-api、heartbeat、ossim-agent、alienvault-api-core、snort、redis-server、ossim-server 等文件。下面以 apache2 配置文件为例，讲解配置文件中的含义，如图 7-12 所示。

```
alienvault:/etc/logrotate.d# cat apache2
/var/log/apache2/*.log {
        weekly
        missingok
        rotate 52
        compress
        delaycompress
        notifempty
        create 640 root adm
        sharedscripts
        postrotate
                /etc/init.d/apache2 reload > /dev/null
        endscript
}
alienvault:/etc/logrotate.d# _
```

每周轮询
指定的日志在删除前，转储52次
采用Gzip压缩
指定创建日志权限640，Owner是root，Group为adm
在转储之后需要执行重新载入apache2的命令

图 7-12 Apache 日志轮询配置及解释

Rotate的值为52，代表该设置在从系统中删除日志文件之前，将保留最后52个消息；weekly表示日志将每周轮转一次。Logrotate 中的其他选项是每月和每天轮转。这个脚本的启动由/etc/cron.daily/logrotate 下的计划任务负责执行。

7.2.11 OSSIM 分布式系统中日志存储问题

本书第 2 章详细讲解了 OSSIM 分布式系统架设，这里将分析这种系统下的日志存储问题，根据图 2-37（OSSIM 分布式安装）所示，有以下两种日志存储方案：

（1）网络应用及交换机、路由器日志，转发日志到它们所在 VLAN 的 Sensor 中，那么其 RAW LOG 会存储在 Sensor 端，而标准化处理后的事件会发送到 OSSIM Server 端，经过关联分析后，结果保存到 Server 端 MySQL 数据库中，这样所有 RAWLOG 会分散在多个 Sensor 中。如果一旦 Sensor 宕机出现了数据丢失，则原始日志不保。

（2）将网络应用 IISapache 交换机等日志发送到 OSSIM Server 端，专门负责收集的一块网卡中，例如 OSSIM 端有三块网卡:

- eth0：管理接口（IP1）
- eth1：嗅探口（无 IP 地址）
- eth2：日志接收口（IP2）

那么日志将转发到 eth2 口，全网 RAW Log 数据集中存储在/var/log/alienvault/device/目录，与此同时经插件归一化处理后的事件会存储在 alienvault_siem 库中。

7.3 日志协议 Syslog

完善的日志分析系统应该能够通过多种协议（包括 Syslog 等）进行日志采集并对日志进

行分析。因此，日志分析系统首先需要实现对多种日志协议的解析。其次，需要对收集到的海量日志信息进行分析，再利用数据挖掘技术，发现隐藏在这些日志中的安全问题。本章将介绍日志采集中的各种协议及使用方法。

Syslog 在 UNIX 系统中应用广泛，它是一种标准协议（RFC3164），负责记录系统事件（event）的一个后台程序，记录内容包括核心、系统程序的运行情况及所发生的事件。Syslog 协议使用 UDP 作为传输协议，通过 514 端口通信，Syslog 使用 syslogd 后台进程，syslogd 启动时读取配置文件/etc/syslog.conf，它将网络设备的日志发送到安装了 Syslog 软件系统的日志服务器，Syslog 日志服务器自动接收日志数据并写到指定日志文件中。

7.3.1 常见日志收集方式

在网络管理中用来采集日志数据的方式包括：文本方式采集、SNMP Trap 方式采集，以及 Syslog 协议收集。

1. 文本方式

以文本方式采集日志数据主要是指通过 SMB 方式共享，或者邮件发送，或者以 FTP 上传方式来收集设备日志数据。这几种方式的共同点主要以原始日志为主，不能对日志进行深度分析。随着大数据时代的到来，日志的产量越来越大，所以这种方式不适应今后的网络发展，但是分析原始日志的基本方法需要大家掌握。

在图 7-13 中显示了 Squid 代理服务器原始日志的格式，由于冗长且不便阅读，人们通常会把 Raw Log 经过预处理后再显示。

图 7-13 原始日志格式

格式化处理的日志相当规整，符合人们的阅读习惯，方便查找问题，如图 7-14 所示。

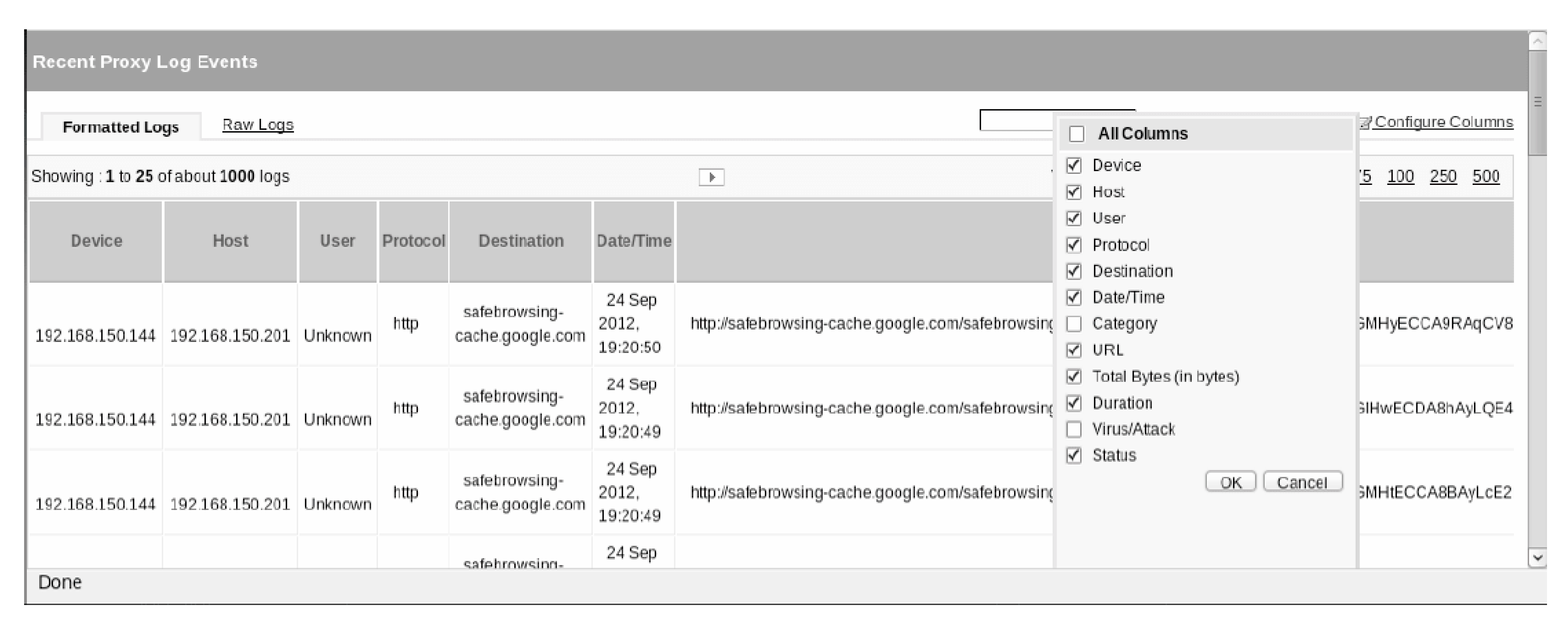


图 7-14　处理过后的日志

2. SNMP Trap 方式

对于建立在 SNMP（Simple Network Management Protocol）上的网络管理，可以通过 SNMP Trap 方式来进行日志数据的采集。Trap 指的是被管理设备向 SNMP 管理者发送的通告网络的陷阱报文，比如设备的冷热启动，一些端口状态的不可用，以及用户登录失败等事件。在网络中设备的日志，通过对 SNMP 数据报文中 Trap 值的解读，就可以获得关于该网络设备的故障信息。

3. Syslog 方式

Syslog 协议应用在服务器、路由器以及交换机等网络设备中，Syslog 能够记录系统中发生的各种事件和活动，而且 Syslog 能够以远程的方式接收来自设备的日志记录，能够把来自多个设备的日志记录以文件形式进行保存，用这种方法收集日志在企业应用中最为常见。

7.3.2　日志的标准化

由于不同网络设备所产生的日志格式不同，这就需要我们将不同格式的日志转化为统一格式。例如记录日期，有些系统的日志使用 mm/dd/hr:mm:ss 格式，有些会使用 24 小时的记录方式。标准化就需要将这些时间记录格式转化为统一的格式。标准化不仅限于时间格式的统一，而要将所有日志的格式尽可能统一，以便数据的查询处理。

7.3.3　主流日志格式介绍

目前主流网络设备所产生的日志格式主要有三种：Syslog，Traffic Log 和国际通行的 WELF（WebTrends Enhanced Log Format）。一个 Syslog 消息完整格式由三个可识别的部分所组成：

- PRI：优先级 Priority
- HEADER：报头
- MSG：消息描述

Syslog 消息并没有对最小长度有所定义，但报文的总长度必须在 1024 字节之内。其中 PRI 部分必须有 3 个字符，以“<”为起始符，然后紧跟一个数字，最后以“>”结尾。在括号内的数字被称为 Priority（优先级），Priority 值由 Facility 和 Severity 两个值计算得出，这两个值的级别和含义见表 7-1 和表 7-2。大家先看个例子：

```
<30>Oct 10 20:30:10 fedora auditd [1780]: The audit daemon is exiting
```

其中<30>是 PRI 部分，即 Priority，取值范围 0~191。

“Oct 10 20:30:10 fedora”是 HEADER（报头）部分。

“auditd [1780]: The audit daemon is exiting”是 MSG（消息描述）部分。

在 PRI 部分，该数值和 Facility、Level 有关 ，Facility 是创建日志的实体，比如由 Kernel 产生，还是 User 产生，或者是 Mail 产生……而 level 可以看成是日志的级别。他们的关系可以利用公式推导 ：

$$Priority=Facility\times 8 + Level \qquad （公式 7-1）$$

后台监控程序会被分配一个 facility 值，而没有被分配到 facility 值的进程则会使用“local user”的 facilities 值，比如很多网络设备都会默认使用 facility 值“local user 7”来发送信息。对于“SYSLOG　Facility”的理解，大家可以参考 7.17.2 节中有关 Snare 配置的相关内容。

表 7-1　定义 Facility 级别

编号	Facility	编号	Facility
0	Kernel Messages（内核生成的日志消息）	12	NTP subsystem（NTP 日志消息）
1	User-level messages（用户日志消息）	13	Log audit
2	Mail system（邮件系统日志消息）	14	Log alert
3	System daemons（系统守护进程消息）	15	Clock daemon
4	Security/authorization messages（安全管理日志消息）	16	localuse 0（系统保留）
5	Messages generated intermally by syslogd syslog（自己的日志消息）	17	localuser1
6	Line printer subsystem（打印机日志消息）	18	localuser2
7	Network news subsystem（新闻服务日志消息）	19	localuser3
8	UUCP subsystem（UUCP 日志消息）	20	localuser4
9	Clock daemon（时钟进程）	21	localuser5
10	Security/authorization messages	22	localuser6
11	FTP daemon（FTP 日志消息）	23	localuser7

从分类能看出来 Syslog 的 Facility 有一部分（序号 16~23）是为其他程序预留的，例如 Cisco 设备使用 local4 发送 PIX 防火墙的 Syslog 日志。

表 7-2　严重等级描述

编号	Severity	编号	Severity
0	Emergency（系统不可用）	5	Notice（具有重要性的普通事件）
1	Alert（必须马上采取措施）	6	Informational（有用事件）
2	Critical（关键事件）	7	Debug（调试信息）
3	Error（错误事件）	8	除了 none 之外的所有级别
4	Warning（警告事件）	9	None 没有优先级，用于排错

7.3.4　Syslog 日志记录级别

由于 Syslog 反映系统底层的诸多信息，它记录的信息非常全面，并把日志分为 8 种优先级，如图 7-15 所示。Syslog 记录的系统事件有（系统、网络、高可用等，每种事件都分为 0～7 级）：

- 系统内核产生的 0～7 级，包括相关的硬件问题。
- 网络部分产生的 0～7 级。
- 高可用性部分产生的 0～7 级。
- 设备驱动程序的 0～7 级。
- 各类系统 daemon 产生的 0～7 级，如 SNMP 模块。
- 系统服务模块，如 WWW、DNS、MAIL、Squid、各种防病毒软件。
- 第三方和应用系统，如 Tripwire、TCP Wrapper、Snort、CheckPoint 等。
- 系统管理过程中产生的 syslog。
- 用户写在/etc/services 中的服务程序。
- 用户开发程序使用 Syslog API 产生的日志。

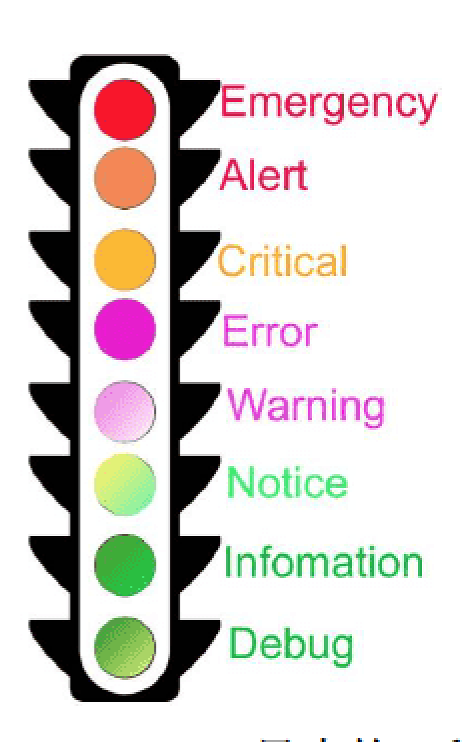


图 7-15　SYSLOG 日志的 8 种优先级

7.3.5　Syslog.conf 配置文件

Syslog.conf 是 syslogd 进程的配置文件，在程序启动时读取，其格式为 facility.level，由上面两张表中的 facility 和 Severity 所组成之间用点分隔。下面以 CentOS 5.x 为例讲解 Syslog.conf

配置文件中关键语句的作用，如图 7-16 所示。

```
#cat /etc/syslog.conf
```

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure
# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog
local4.*      /var/log/cisco.log
# Log cron stuff
cron.*                                                  /var/log/cron
# Everybody gets emergency messages
*.emerg                                                 *
# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler
# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

图 7-16　syslog 配置

下面解释一下这个配置的含义：

（1）*.info;mail.none;authpriv.none /var/log/messages

含义：将 info 或更高级别的消息送到/var/log/messages，除了 mail 以外，其中*是通配符，代表任何设备；none 表示不对任何级别的信息进行记录。

（2）authpriv.* /var/log/secure

含义：将 authpriv 设备的任何级别的信息记录到/var/log/secure 文件中，这主要是一些和认证、权限使用相关的信息。

（3）mail.*　/var/log/maillog

含义：将 mail 设备中的任何级别的信息记录到/var/log/maillog 文件中，这主要是和电子邮件相关的信息。有的配置文件会这样写：

```
mail.* -/var/log/maillog
```

表示邮件产生的信息不直接存入该文件，而是先存在缓存中，也就是不对文件系统执行 sync，等到信息量达到一定程度后，再存储磁盘，所以要注意正常关机，否则会影响该文件的完整性。

（4）cron.* /var/log/cron

含义：将 cron 设备中的任何级别的信息记录到/var/log/cron 文件中，这主要是和系统中定期执行的任务相关的信息。

（5）*.emerg　　　　*

含义：将任何设备的 emerg 级别的信息发送给所有正在系统上的用户。

（6）local7.* /var/log/boot.log

含义：将和系统启动相关的信息记录到/var/log/boot.log 文件中。

更多信息可采取 man syslog.conf 的方式查询。

7.3.6 用 Tcpdump 分析 Syslog 数据包

为了让大家清楚了解远程传输的 syslog 消息内容，我们用以下命令查看，首先确定 OSSIM 中防火墙允许 syslog 包通过，然后执行：

```
#tcpdump -Xni eth0 port 514
```

```
VirtualUSMAllInOne:~/.ansible/tmp# tcpdump -Xnieth0 port 514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
02:38:01.681921 IP 192.168.11.89.47184 > 192.168.11.105.514: SYSLOG authpriv.info, length: 107
        0x0000:  4500 0087 0000 4000 4011 a253 c0a8 0b59  E.....@.@..S...Y
        0x0010:  c0a8 0b69 b850 0202 0073 1552 3c38 363e  ...i.P...s.R<86>
        0x0020:  4d61 7220 2032 2031 323a 3235 3a30 3120  Mar..2.12:25:01.
        0x0030:  616c 6965 6e76 6175 6c74 2043 524f 4e5b  alienvault.CRON[
        0x0040:  3230 3031 305d 3a20 7061 6d5f 756e 6978  20010]:.pam_unix
        0x0050:  2863 726f 6e3a 7365 7373 696f 6e29 3a20  (cron:session):.
        0x0060:  7365 7373 696f 6e20 6f70 656e 6564 2066  session.opened.f
        0x0070:  6f72 2075 7365 7220 726f 6f74 2062 7920  or.user.root.by.
        0x0080:  2875 6964 3d30 29                        (uid=0)
```

7.3.7 Syslog 的安全漏洞

尽管 Syslog 协议在网络日志的管理方面做得非常优秀，提供了跨平台的日志传输通道和日志存储策略。如果使用 Syslog Server 进行日志集中管理，Syslog 使用 UDP 协议进行打包传送的数据就难辨真伪，从而使得 Syslog 协议失去意义而彻底崩溃。Syslogd 是以明文的形式传送数据的，入侵者用 tcpdump 之类的网络工具可以轻而易举地获取传送数据。但是在信息安全等方面仍存在着漏洞。

Syslog 程序以明文形式存储数据，入侵者可以从/var/log/下获取这些数据。当然这需要拥有 root 权限。一旦入侵者获得 root 权限，就可以肆意篡改/var/log 或/var/adm 下的文件，并删除入侵记录，而这么做不会留下痕迹。

如果攻击者闯入 Linux 系统后，除了修改日志以外，还有可能通过一条“kill all syslogd”命令而结束 Syslog 守护进程，这样会阻止日志的记录，即便是有远程日志收集系统也无济于事。如果系统管理员将 syslogd 重命令为其他名称，从而可以躲避这种攻击。

7.3.8 配置 SNMP

下面讲解配置 SNMP 的几个主要步骤：

（1）在 OSSIM 中启动 SNMP。

（2）编辑/etc/ossim/ossim_setup.conf。

将 snmpd=no 修改成 snmpd=yes ，snmptrap=no 修改成 snmptrap=yes。

（3）测试 SNMP。

①本机测试 SNMP

```
#netstat -na |grep 161
udp        0      0 0.0.0.0:161            0.0.0.0:*
```

测试环境采用 community 字符串为 public。

```
#snmpwalk -v 1 -c public loclahost:161
```

下面会显示很长输出：

```
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = OID: iso.3.6.1.6.3.1.1.5.3
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = OID: iso.3.6.1.6.3.1.1.5.4
.....
```

如果获取到大量数据，说明 SNMP 服务成功启动。

②远程主机测试

先关闭防火墙。

```
#snmpwalk -v 1 -c public 192.168.150.114:161
```

在 OSSIM 系统里 SNMP 日志记录在/var/log/daemon.log 文件中。例如：

```
Feb  7 23:09:48 localhost snmpd[25169]: Connection from UDP: [127.0.0.1]:43328->[127.0.0.1]
Feb  7 23:09:48 localhost snmpd[25169]: Connection from UDP: [127.0.0.1]:43328->[127.0.0.1]
Feb  7 23:09:48 localhost snmpd[25169]: Connection from UDP: [127.0.0.1]:43328->[127.0.0.1]
Feb  7 23:12:15 localhost snmpd[25169]: Connection from UDP: [192.168.150.90]:57943->[192.168.150.114]
Feb  7 23:12:15 localhost snmpd[25169]: Connection from UDP: [192.168.150.90]:57943->[192.168.150.114]
Feb  7 23:12:15 localhost snmpd[25169]: Connection from UDP: [192.168.150.90]:57943->[192.168.150.114]
…
```

7.4 原始日志格式对比

原始日志可能是一个通用的系统消息、应用程序日志、SNMP Trap，但是这些日志格式不统一，无法做关联分析，OSSIM 需要将它们输出成统一格式，下面我们先看个例子。

举例：我们先查看一条 SSH 的日志也是原始日志格式。主机 server-1，/var/log/auth.log 文

件显示内容如下：

```
   May 30 13:15:52 server-1 sshd[13980]: Accepted pasword for root from
192.168.150.20 port 4545 ssh2
```

经过归一化日志处理后会发生什么变化呢？这些字段可以从日志消息填充。处理后的日志也就是在 OSSIM Sensor 机器中的日志/var/log/ossim/agent.log，内容如下：

```
   2013-05-30 13:15:49,441 Output[INFO]:event type="detector" date="1275239780"
sensor="192.168.150.201" interface="eth0" plugin_id="4003" plugin_sid="7"
src_ip="192.168.150.20" src_port="4545" dst_ip="192.168.150.200" dst_port="22"
username="root" log="May 30 13:52 server-1 sshd[13980]:Accepted password for root
from 192.168.150.20 port 4545 ssh2" fdate="2013-05-30 13:15:52" tzone="8.0"
```

从内容上看起来比原来复杂了许多，但这是为了统一标准格式。它会传送至 Server 的 EDB 数据库中统一存储。

我们再看看 OSSIM 服务器对于这样的事件都做了哪些处理呢？这包括特定的类型和子类型以及该资产的值。OSSIM Server 上/var/log/alienvault/server.log 内容如下：

```
   2013-05-30 06:48:41 OSSIM-Message: Event received: event id="0" alarm="0"
type="detector" fdate="2013-05-30 13:15:52" date="1275239780"
tzone="8.0"plugin_id="4003" plugin_sid="7" src_ip="192.168.150.20"
src_port="4545" dst_ip="192.168.150.200" dst_port="22" sensor="192.168.150.201"
interface="eth0" protocol="TCP" asset_src="2" asset_dst="2" log="May 30 13:15:52
server-1sshd[13980]: Accepted password for root from 192.168.150.20 port 4545 ssh2"
username="root"
```

随后，系统通过 Web 界面展示在前台的控制台上，用户可以通过访问 SIEM 查看。

7.5 插件配置步骤

经过以上描述，大家了解收集日志的流程，接下来就要建立脚本，步骤如下：

（1）新建插件文件，通常复制一个现有的脚本文件，并修改其内容，以符合新的应用程序需求。

（2）定义一个通用规则，这是最后的规则，它捕获所有的事件，不能根据特定规则进行分组。

（3）去除噪声，OSSIM 可以排除某些无关事件子类型的事件，这些被视为噪声，说得简单点就是在 IDS/IPS 等安全设备上产生的海量重复报警就是噪声。

（4）通过 OSSIM 代理注册插件，为了将事件发送到 OSSIM 服务器，需要将插件激活，插件的路径必须在代理配置文件中指定。

（5）通过 OSSIM Server 注册插件，以让服务器知道事件的优先级和可靠性价值的事件，就必须在 Server 端也注册插件。

（6）在 Server 端激活插件，重启 OSSIM Server 进程。

```
#/etc/init.d/ossim-server restart
```

（7）在 Agent 代理端激活插件，重启 OSSIM Agent 进程。

```
#/etc/init.d/ossim-agent restart
```

7.6 插件导入

假设有一段导出的 SQL 文件，其中包含有可执行 SQL 语句。例如将 MySQL 数据库备份到 test.sql 文件里，就可以用下面方法进行还原：

```
#mysql < test.sql
```

可以在 MySQL 的提示符下，用 SOURCE 命令来加载 SQL 文件。但如果压缩了 SQL 文件怎么做还原呢？可以先解压缩再加载，命令如下：

```
#gunzip -c test.sql.gz |mysql
```

OSSIM 在安装后期通过一些 SQL 语句集中导入插件，导入完毕，放置在 /usr/share/doc/ossim-mysql/contrib./plugins/目录下，扩展名为 sql.gz。如果发现某些插件需重新导入数据库，可以先用 gunzip 命令解压 sql.gz 文件，再使用“ossim-db <file.sql”方式导入。如果是新插件怎么办？就复制一个功能类似插件，然后修改 SQL 代码，再导入数据库。那如果只还原单独的表（例如表 asset）又会怎样？看看如下操作：

```
#grep 'INSERT INTO `asset`' test.sql |mysql test
```

或者文件是压缩的：

```
#gunzip -c test.sql.gz |grep 'INSERT INTO `asset`'|mysql test
```

test 代表实例数据库名称。一旦 MySQL 加载完数据，gunzip 就会自动退出。

7.7 插件注册操作实例

Linux 系统中 last 命令会读取位于/var/log 目录下的 wtmp 文件，并把该文件的内容记录的登录系统的用户名单全部显示出来。状态更新将被系统用 logger 命令记录。本实验以 OSSIM 4.3 为平台。

（1）通过以下脚本来监控 last 的状态

```
#!/bin/sh
# create the file if does not exist
```

```
    touch /var/log/last.prev
    while true
    do
    # get last entries
    last > /var/log/last.new
    # send new entries to syslog
    diff /var/log/last.prev /var/log/last.new | grep '^>' | logger -t LOGON_EXAMPLE
-p local2.info
    # move .new to .prev
    mv /var/log/last.new /var/log/last.prev
    sleep 5
    done
```

（2）日志样本

```
    # tail -f /var/log/messages
    Jun 10 20:21:32 server-1 LOGON_EXAMPLE: > root pts/3 localhost Wed Jun 10 18:49
- 20:21 (00:31)
    Jun 10 20:23:28 server-1 LOGON_EXAMPLE: > dbadmin pts/3 localhost Wed Jun 10
20:23 still logged in
    Jun 10 20:23:59 server-1 LOGON_EXAMPLE: > root pts/4 localhost Wed Jun 10 20:23
still logged in
    Jun 10 20:24:09 server-1 LOGON_EXAMPLE: > root pts/4 localhost Wed Jun 10 20:23
- 20:24 (00:00)
    Jun 10 20:24:09 server-1 LOGON_EXAMPLE: > dbadmin pts/3 localhost Wed Jun 10
20:23 - 20:24 (00:00)
    Jun 10 20:24:09 server-1 LOGON_EXAMPLE: > root pts/2 192.168.150.20 Wed Jun 10
18:38 - 20:24 (00:45)
    Jun 10 20:24:54 server-1 LOGON_EXAMPLE: > root pts/2 192.168.150.20 Wed Jun 10
20:24 still logged in
    Jun 10 20:26:15 server-1 LOGON_EXAMPLE: > root pts/2 192.168.150.20 Wed Jun 10
20:24 - 20:26 (00:01)
    Jun 10 20:26:20 server-1 LOGON_EXAMPLE: > ossim pts/2 192.168.150.20 Wed Jun
10 20:26 still logged in
    Jun 10 20:26:25 server-1 LOGON_EXAMPLE: > ossim pts/2 192.168.150.20 Wed Jun
10 20:26 - 20:26 (00:00)
```

（3）在 OSSIM Agent 上修改 rsyslog.conf

```
    #vi /etc/rsyslog.conf
```

在该文件末尾加入以下内容：

```
    local2.info /var/log/last_logon.log
```

（4）重启 rsyslog 服务

```
    #/etc/init.d/rsyslogd restart
```

检查是否有新的条目写入新日志文件。

```
    plugins# tail -f /var/log/last_logon.log
    Jun 10 19:38:49 server-1 LOGON_EXAMPLE: > root pts/2 localhost Wed Jun 10 19:38
still logged in
    Jun 10 19:38:54 server-1 LOGON_EXAMPLE: > root pts/2 localhost Wed Jun 10 19:38
- 19:38 (00:00)
    Jun 10 19:38:59 server-1 LOGON_EXAMPLE: > ossim pts/2 localhost Wed Jun 10 19:38
still logged in
    Jun 10 19:40:51 server-1 LOGON_EXAMPLE: > ossim pts/2 localhost Wed Jun 10 19:38
- 19:40 (00:01)
```

```
    Jun 10 20:15:09 server-1 LOGON_EXAMPLE: > reboot system boot 2.6.31.6 Wed Jun
10 17:39 - 20:15 (02:35)
```

（5）新建插件文件

```
#cd /etc/ossim/agent/plugins
#cp syslog.cfg myexample.cfg
```

（6）修改新插件参数

```
;; Building Plugins MyExample
;; plugin_id: 9001
;; type: detector
[DEFAULT]
plugin_id=9001
[config]
type=detector
enable=yes
source=log
# Enable syslog to log everything to one file. Add it to log rotation also.
# echo "*.* /var/log/all.log" >> /etc/syslog.conf; killall -HUP syslogd
#location=/var/log/all.log
location=/var/log/last_logon.log
... ...
```

（7）在 OSSIM Agent 上注册插件

编辑/etc/ossim/agent/config.cfg 文件。

在[plugins]中加入插件，如图 7-17 所示。

```
myexample=/etc/ossim/agent/plugins/myexample.cfg
```

最后打开 ossim-setup 配置程序并选择 Configure Sensor→Select Data Sources，找到 myexample 插件选中后，保存退出。

图 7-17　新注册插件

（8）在 OSSIM Server 端注册插件

首先复制现有的 SQL 脚本来建立新的数据结构：

```
#cd /usr/share/doc/ossim-mysql/contrib/plugins
#cp syslog.sql myexample.sql
```

如果是 syslog.sql.gz 需要先解压，接下来获取列表插件配置文件中定义的规则。

```
#grep '^\[' /etc/ossim/agent/plugins/myexample.cfg
[DEFAULT]
[config]
[Rule 01 - Console Session Open]
[Rule 02 - Console Session Closed]
[Rule 03 - New User Session - IP]
[Rule 04 - New User Session - hostname]
[Rule 05 - User Session Closed - IP]
[Rule 06 - User Session Closed - hostname]
[Rule 07 - Reboot Detected]
```

具有相同的 plugin_sid 规则需要一条 SQL 语句以及在 plugin_sid 中定义服务器，如不同规则的，则会由 last 返回源的 IP 或主机名。此时我们需要将 myexample.sql 导入到数据库，使用如下命令：

```
#cd /usr/share/doc/ossim-mysql/contrib/plugins
#cat myexample.sql |ossim-db
```

如果有报错提示，请检查 SQL 语句是否有错误。这时可以在 Web 界面下的数据源中查看这个插件，如图 7-18、图 7-19 所示。

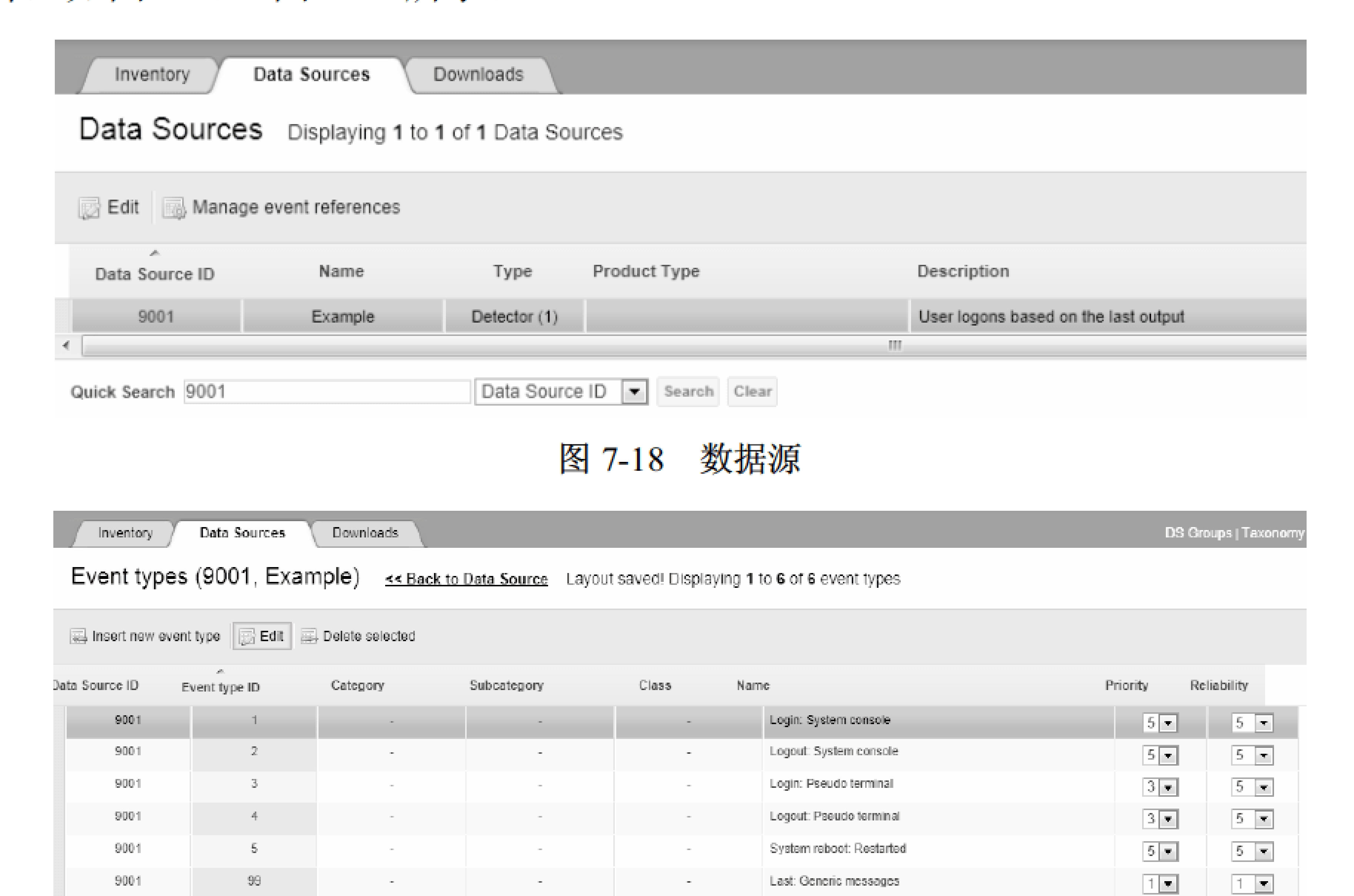


图 7-18　数据源

图 7-19　添加新数据源

当看到上面这些信息时说明插件已成功添加，接下来需要重启服务即可生效。

```
#/etc/init.d/ossim-server restart    \\重启 OSSIM Server
#/etc/init.d/ossim-agent restart     \\重启 Agent
```

最后可以到 SIEM 控制台下查看到日志。

7.8 Agent 插件处理日志举例

7.8.1　收集与处理过程

本节以 Syslog 插件为例说明日志收集与处理的过程。我们需要收集哪种类型的服务，对应着需要在 Sensor 的配置界面中添加相应的插件，如图 7-20 所示。

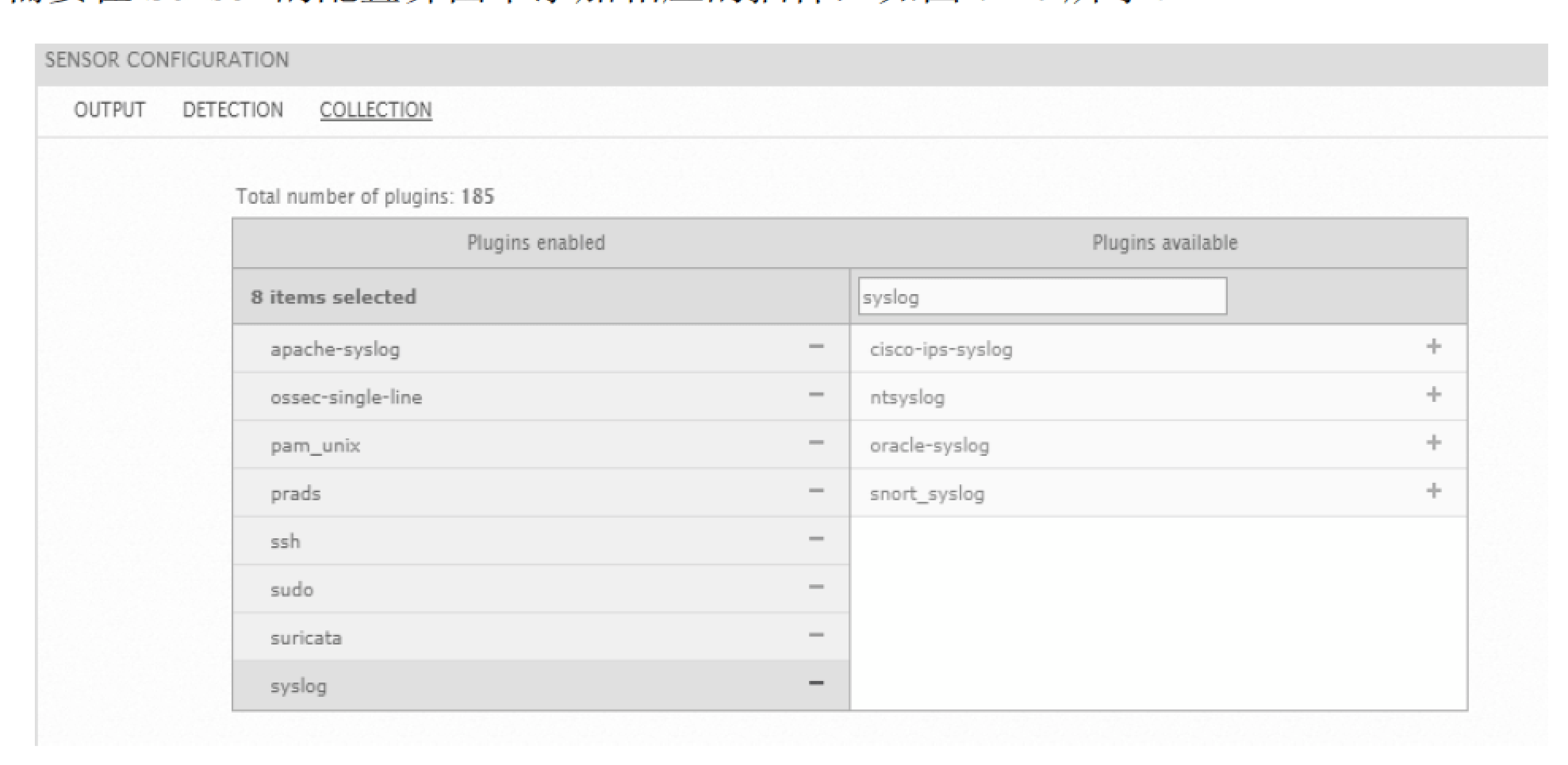


图 7-20　更新插件

当选取某个插件，例如 syslog 插件后，单击应用按钮，稍后会自动在/etc/ossim/ossim_setup.conf 配置文件的[sensor]域的 detectors 选项中添加 syslog。如图 7-21 所示。添加的插件是否成功呢?我们可以通过查询 agent.log 日志实现，具体操作说明如下。

```
[sensor]
asec=no
detectors=pam_unix, ssh, prads, apache-syslog, syslog, ossec-single-line, sudo, suricata
ids_rules_flow_control=yes
interfaces=eth0
ip=
"/etc/ossim/ossim_setup.conf" 84L, 1608C
```

图 7-21　ossim_setup 配置文件中的插件列表

比如，我们在“Plugins avaliable”中添加了 iptables 插件，添加成功后日志会记录到/var/log/alienvault/agent/agent.log 文件中。

```
    #cat  /var/log/alienvault/agent/agent.log |grep iptables
    Feb 27 00:44:23 Virtual python: Alienvault-Agent[INFO]:WATCHDOG
-plugin(iptables) is enabled
```

凡是修改 ossim_setup.conf 配置文件后，需要运行 ossim-reconfig 命令使设置生效。

OSSIM 中 syslog 插件收集的系统日志，如图 7-22 所示。

Time frame selection GMT-4:00: Last 2 Hours | Last 24 Hours | Last Week | Last Month | All | Custom　　About 0 logs

SHOW 50 ENTRIES　　TIME RANGE: 2014-09-25 22:15:31 <-> 2014-09-26 00:15:31 GMT-4:00

ID	DATE GMT-4:00	TYPE	SENSOR	SOURCE	DESTINATION	DEVICE	DATA		SIGNATURE
1	2014-09-26 00:15:23	syslog	VirtualAllInOne6x1GB	10.32.14.133	0.0.0.0	10.32.14.133	Sep 26 00:15:23 VirtualAllInOne6x1GB postfix/qmgr[4265]: 2354F2EC49BB: removed	[...]	Validate

EVENT TYPE	EVENT DETAIL
Syslog: syslog entry	Sep 26 00 15 23 VirtualAllInOne6x1GB postfix/qmgr[4265] 2354F2EC49BB removed

SENSOR	PRODUCT TYPE	CATEGORY	SUB-CATEGORY
VirtualAllInOne6x1GB	Operating System	System	Information

USERDATA1	USERDATA2	USERDATA3	USERDATA4	USERDATA5	USERDATA6	USERDATA7	USERDATA8	USERDATA9
1d4cbaae1013db27ec6901edb72eacd6	Sep 26 00:15:23 VirtualAllInOne6x1GB postfix/qmgr[4265]: 2354F2EC49BB: removed	2354F2EC49BB: removed	empty	empty	empty	empty	empty	empty

ID	DATE GMT-4:00	TYPE	SENSOR	SOURCE	DESTINATION	DEVICE	DATA		SIGNATURE
2	2014-09-26 00:15:23	syslog	VirtualAllInOne6x1GB	10.32.14.133	0.0.0.0	10.32.14.133	Sep 26 00:15:23 VirtualAllInOne6x1GB postfix/local[13517]: 2354F2EC49BB: to=<root@VirtualAllInOne6x1GB>, orig_to=<root>, relay=local, delay=4, delays=3.8/0.05/0/0.1, dsn=2.0.0, status=sent	[...]	Validate

图 7-22　插件收集 syslog 日志

网络设备或应用将日志以 syslog 形式发给 Sensor，在探针中含有 ossim-agent，以实现归一化处理，这样日志就会存在/var/log/目录下的某个文件中，例如 Sensor 的 IP 地址为 192.168.11.105，我们在 Buffalo 设备的日志收集设置中的设置如图 7-23 所示：

图 7-23　指向日志收集服务器

通过以上设置可将 Buffalo 防火墙日志转发到 Sensor（192.168.11.105）所在的 OSSIM 系统中，由 ossim-agent 处理防火墙收集的日志，并存放在/var/log/syslog 文件，部分内容如下：

```
May  3 10:45:33 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:1057 to 192.168.117.176:445 (br0
May  3 10:45:33 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:1058 to 192.203.120.2:445 (br0)
May  3 10:45:34 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:1059 to 188.247.82.151:445 (br0)
May  3 10:45:35 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2081 to 2.58.137.158:445 (br0)
May  3 10:45:35 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2067 to 220.251.221.40:445 (br0)
May  3 10:45:36 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2086 to 26.81.153.210:445 (br0)
May  3 10:45:36 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2070 to 192.51.207.15:445 (br0)
May  3 10:45:37 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2065 to 192.219.91.145:445 (br0)
May  3 10:45:37 192.168.11.1 [4C: E6:76:43:2D:E7] buff : FIREWALL: TCP connection denied from 192.168.11.223:2089 to 179.47.61.137:445 (br0)
```

下一步，启用 syslog 插件，如图 7-20 所示。当选中 syslog 插件后，在/etc/ossim/agent/config.cfg 配置文件中就会添加一条语句，接下来输入以下命令查看效果。

```
alienvault:~# cat /etc/ossim/agent/config.cfg|grep syslog.cfg
apache-syslog=/etc/ossim/agent/plugins/apache-syslog.cfg
ntsyslog=/etc/ossim/agent/plugins/ntsyslog.cfg
syslog=/etc/ossim/agent/plugins/syslog.cfg
alienvault:~#
```

此时 ossim-agent 会调用/etc/ossim/agent/plugins/下面对应的 syslog.cfg 插件，来分析/var/log/syslog 对应的日志。那技术上如何实现呢？通过正则表达式来提取日志中的相关信

息，syslog 中正则表达式如下：

```
regexp="^(?P<logline>(\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(
?P<generator>[^\[]*)\[(?P<pid>\d+)\]:(?P<logged_event>.*))$"
device={resolv($sensor)}
date={normalize_date($1)}
plugin_sid=1
userdata1={md5sum($logline)}
userdata2={$logline}
userdata3={$generator}
userdata4={$logged_event}
userdata5={$pid}
```

最后，ossim-agent 将会把这些字段内容发给 Server，Server 再继续处理。下面是关于 Sensor 上运行插件的使用案例。

举例：收集 Apache 访问日志，可以通过 apache-syslog 插件解决这一问题。在 Apache Server 上需要修改 http.conf 配置文件，并加入下面内容。

```
CustomLog "|/usr/bin/logger -t httpd -p local6.info" combined
```

在 rsyslog.conf 配置文件加入一行：

```
local6.* @AlienVault IP
```

AlienVault IP 推荐为 OSSIM Server 上日志收集网卡的 IP 地址。

当修改了插件内容，则必须重启 ossim-agent 服务。注意：在 /usr/share/doc/ossim-mysql/contrib/plugins/下，可以找到和插件相关的所有 SQL 文件。

7.8.2　常见 Windows 日志转换 syslog 工具

在交换机、路由器、Unix/Linux 中自带 syslog 而 Windows 系统则不带，因为 Windows 系统采用 eventlog 服务记录程序和 Windows 发送事件消息，并不包含 syslog 日志发送机制。如果想收集 Windows 日志，就需要安装 Agent 进行转发，这种软件包括以下几种：

（1）evtsys 的特点是高效、速度快且支持 32 位和 64 位系统。

（2）在 OSSIM 中提供的是 snare，支持 32 位和 64 位系统。通过在客户端上例如 Windows 系统安装 snare 后，在 Windows 系统本机浏览器打开 http://localhost:6161/，在左侧 Network Configuration 中设定 Destnation Snare Server Address 和 Destination port 即可。

对于 syslog 服务器的考虑：一般的小型网络中，可以采用 Winsyslog 或 Kiwisyslog（与之对应的免费工具是 NTsyslog）来建立 syslog 服务器，其部署拓扑如图 7-24 所示，但在大型网络环境中，这种部署方式有些无法胜任。本书 7.17.5 节还会继续讨论 Snare 和 WMI 的区别。

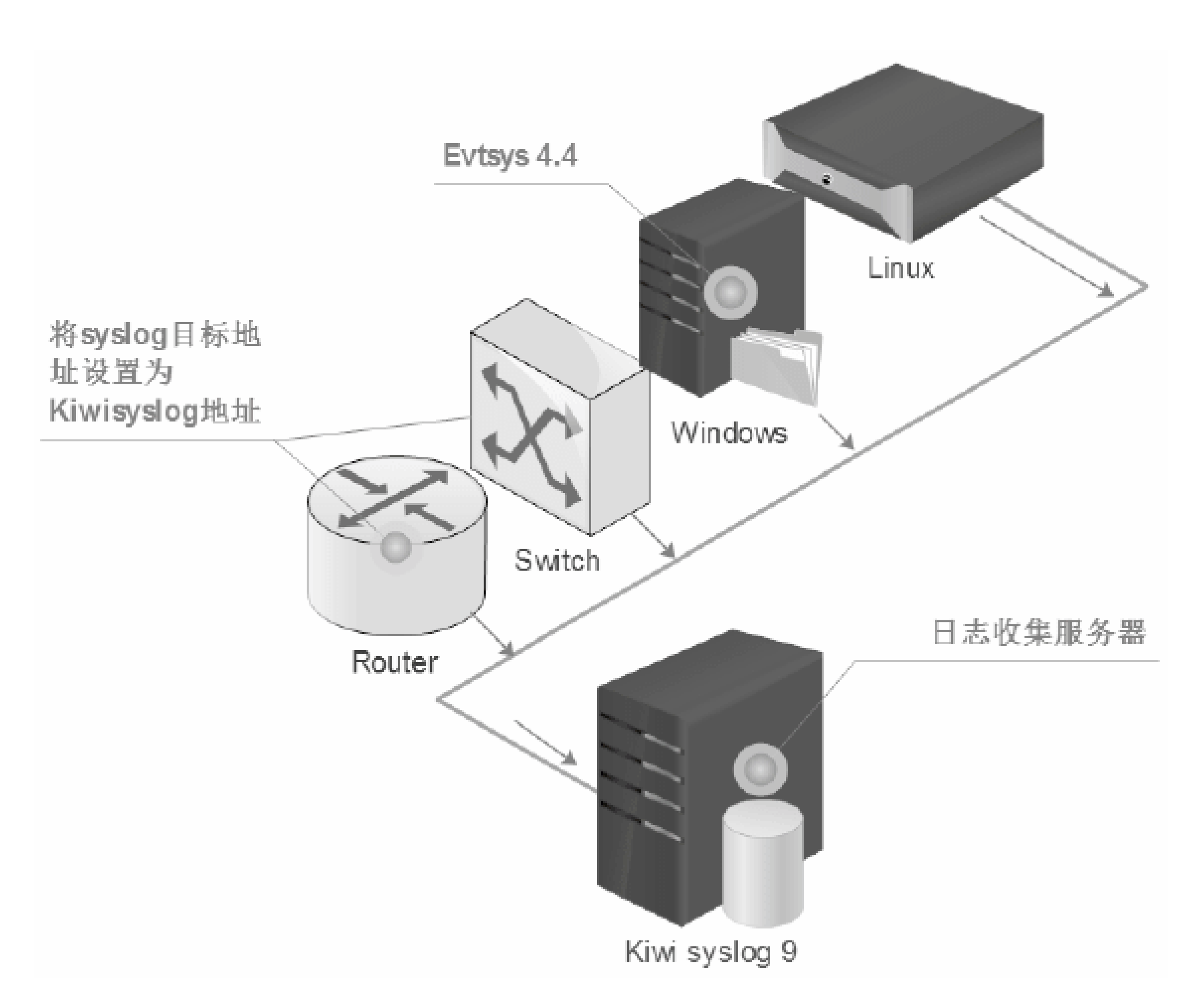


图 7-24　基于 Windows 平台的小型网络日志收集方案

7.8.3　Windows 日志审核

以 evtsys 为例，它通过 UDP 3072 端口，将日志发送给远程的 syslogd 进程。那我们需要收集 Windows 中那些日志？先在 windows 设置→安全设置→本地策略→审核策略中配置以下内容，首先配置 Windows 的本地审核策略，启动本地审核，推荐使用如表 7-3 所示的审核策略。

表 7-3　策略定义

账户管理	成功	失败
登录事件	成功	失败
对象访问		失败
特权使用		失败
系统事件	成功	失败
目录服务访问		失败
账户登录事件	成功	失败
策略更改	成功	失败

7.8.4　收集 Windows 平台日志

Windows 有自己的日志协议，称为 Event Log，Windows 操作系统本身并不支持将日志发送到 syslog 服务器。为解决这一问题可以通过 Agent 或 WMI 也可以通过 evtsys 工具来实现。

在我的博客中下载 evtsys 后，将其复制到系统目录，XP 下是 Windows\system32 目录。然后在 CMD 下执行：

```
C:\>evtsys -i -h 192.168.1.101
```

参数说明：

```
-i 代表安装成 Window 服务
-h 代表 syslog 服务器地址
```

```
-p 代表 syslog 服务器的接收端口,默认端口514可省略
```

启动 evtsys 服务，命令为“net start evtsys”，查看 Windows 的“服务”，发现在原本 Event Log 服务下面增加了一个“Eventlog to Syslog”，并且已经启动。收集到的 Windows 登录日志如图 7-25 所示。

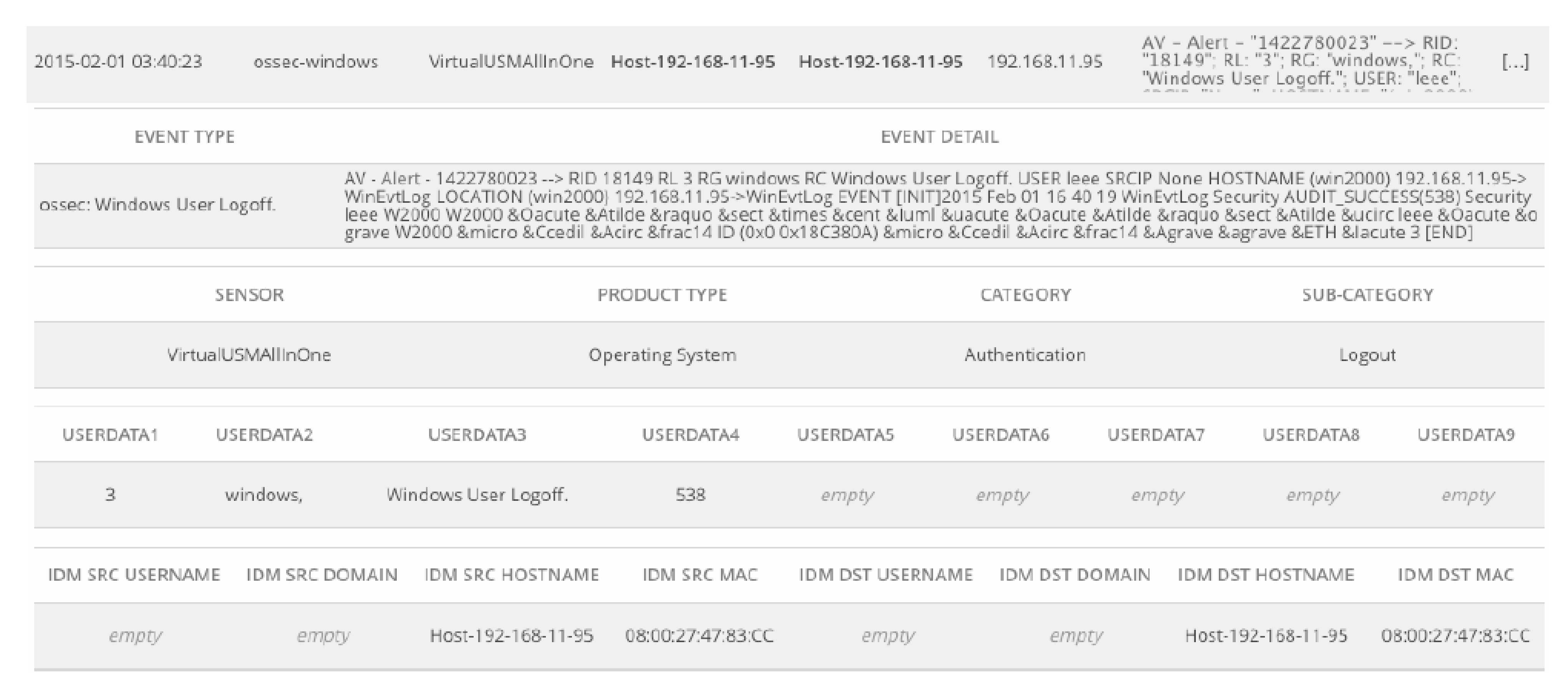


图 7-25　收集的 Windows 登录日志

7.8.5　收集 Cisco 路由器日志

利用添加 syslog 插件相同的方法，选择 cisco-router 插件并启用，我们可以在日志分析中查看到路由器发来的日志信息，如图 7-26 所示。

图 7-26　采集到 Cisco 路由器的一条日志

7.9 Rsyslog 配置

针对 syslog 协议的不足，rsyslog 日志协议应运而生，它提供了丰富的内容过滤和灵活的配置选项、多线程的 syslogd 功能，同一台机器上支持多子 rsyslog 进程，可以监听不同端口。除了继续支持 UDP 传输以外，还增添了使用 TCP 进行传输的功能。

在日志传输安全方面，以前通过 Stunnel 解决了 rsyslog 传输数据加密的问题，目前最新的 rsyslog（版本 8.4.2）版本自身就支持 SSL 加密技术保证安全，在近几年发布的所有 Linux 发行版本中都换成了 rsyslog。

在实际的使用过程中，我们可以通过查看配置文件和相应的日志文件来使用 Rsyslog。在数据库支持方面，它广泛支持各种数据库，尤其对 MySQL 和 Postgres 数据库的支持比较好。

7.9.1 Rsyslog 配置详解

下面简单解释/etc/rsyslog.conf 配置文件的主要内容：

```
#### MODULES ####          #定义日志的模块
$ModLoad imuxsock          #imuxsock 为模块名，支持本地系统日志的模块
$ModLoad imklog            #imklog 为模块名，支持内核日志的模块
#$ModLoad immark           #immark 为模块名，支持日志标记
#$ModLoad imudp            #imupd 为模块名，支持 UDP 协议
#$UDPServerRun 514         #允许514端口接收使用 UDP 和 TCP 协议转发过来的日志
#$ModLoad imtcp            #imtcp 为模块名，支持 TCP 协议
#$InputTCPServerRun 514
#### GLOBAL DIRECTIVES ####            定义全局日志格式的指令
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat #定义日志格式默认模板
$IncludeConfig /etc/rsyslog.d/*.conf   #载入 rsyslog.d 文件中所有以 conf 结尾的文件
#### RULES ####
*.info;mail.none;authpriv.none;cron.none    /var/log/messages
#####记录所有日志类型的 info 级别以及大于 info 级别的信息到/var/log/messages，但是 mail
邮件信息，authpriv 验证方面的信息和 cron 时间#任务相关的信息除外
authpriv.*           /var/log/secure
#####authpriv 验证相关的所有信息存放在/var/log/secure
mail.*              -/var/log/maillog
#####邮件的所有信息存放在/var/log/maillog；这里有一个“-”符号表示是使用异步的方式记录
cron.*               /var/log/cron
####计划任务有关的信息存放在/var/log/cron
*.emerg              *  (*表示所有用户)
###记录所有的≥emerg 级别信息，发送给每个登录到系统的用户
uucp,news.crit       /var/log/spooler
####记录 uucp,news.crit 等存放在/var/log/spooler
local7.*             /var/log/boot.log
####本地服务器的启动的所有日志存放在/var/log/boot.log 中
```

```
###rsyslog.conf 中日志规则定义的格式，参见表7-1
facitlity.priority        Target
#facility: 日志设备(可以理解为日志类型)：
auth                #pam 产生的日志，认证日志
authpriv            #Ssh、Ftp 等登录信息的验证信息，认证授权认证
cron                #时间任务相关
kern                #内核
lpr                 #打印
mail                #邮件
mark(syslog)        #rsyslog 服务内部的信息,时间标识
news                #新闻组
user                #用户程序产生的相关信息
uucp                #unix to unix copy, unix 主机之间相关的通信
local 1~7           #自定义的日志设备
```

从上到下，级别从低到高，记录的信息越来越少，如果设置的日志类型为 err，则日志不会记录比 err 级别低的日志，只会记录比 err 更高级别的日志，也包括 err 本身的日志。

7.9.2　Rsyslog 配置参数含义

在 OSSIM 系统中 rsyslog 有 4 个参数需要大家了解：

- *-m 0*：代表禁用“标记”的消息，只有在-c 参数（代表兼容模式）<3 时使用。
- *-r*：代表启用远程机器的 logging 功能，只有在-c 参数（代表兼容模式）<3 时使用。
- *-x*：代表接收信息时禁用 DNS 查询。
- *-c*：代表兼容模式，-c3，代表运行在 v3 模式，这样开启中心服务器功能必须在 rsyslog.conf 中配置，而不能在命令行设置。关于兼容模式详情通过“man rsyslogd”命令了解。

系统默认/etc/default/rsyslog 参数为 RSYSLOGD_OPTIONS="-c3 -x"。

-x 表示禁止日志服务器解析远程主机的 FQDN（fully qualified domain name）。默认情况下，当有其他机器向自己发送日志消息时，中央日志服务器将尝试解析该机器的 FQDN。如果 Syslog 守护进程无法解析出地址，它会继续尝试，当日志发送量很大时，这种尝试非常浪费时间，应该禁止。

7.9.3　选择合适的日志级别

日志记录器是日志收集的核心部分，我们在客户端收集日志信息包含服务类型，日志级别以及日志接收服务器的 IP 地址。除了 IP 是固定以外，服务类型和日志级别都有多个选项，造成在使用中容易混乱，通常我们在 rsyslog.conf 添加一条信息，用来将日志转发到日志收集服务器，日志格式如下：

服务类型	日志级别	日志接收服务器 IP 地址

在其中服务类型和日志级别又分为若干种，如图 7-27 所示。

主要服务类型
auth, authpriv
认证相关的机制信息
cron
例行性命令相关的信息
kern
与内核相关的信息
mail
与邮件收发相关的信息
news
与新闻用户组服务器相关的信息
user、uucp、local0-local7
与类UNIX机器相关的信息

日志级别
info
notice
warn
err或error
crit
alert
emerg

日志接收服务器IP地址

图 7-27　日志级别的选择

举例：

```
服务类型  日志级别  日志接收服务器 IP 地址
mail.    *    @192.168.100.1
```

这表示表示将 mail 服务的所有级别日志发送至服务器 192.168.0.1，当然有些用户会使用如下配置条目：

```
服务类型  日志级别  日志接收服务器 IP 地址
*.   *   @192.168.100.1
```

7.10 网络设备日志分析与举例

下面分析路由器日志并给出应用实例。网络边界安全中最容易被忽略，但也是最重要的工作是网络日志收集与分析。通过对各种日志文件进行严密监控和分析，来试图识别出入侵和入侵企图，这个过程还涉及在这些日志文件当中对事件进行关联。需要进行检查的网络日志文件有多种类型，还包括设备（包括网络防火墙、路由器）日志。

企业网络设备日志分析分为三类：

- 路由器日志分析
- 防火墙日志分析
- 交换机日志分析

（1）网络日志共同特点

网络设备都可以记录日志，它们对日志记录了的几个核心部分（例如记录网络信息等）分别是：

- 时间戳：包括日期以秒为单位的时间，表示事件发生的时间或事件记录到日志中的时间，有时也标记千分之几秒。
- IP：如源地址、目标地址和 IP 协议（TCP、UDP、ICMP 等）。
- Line 状态和 Line 协议状态：网络设备经常会出现掉电重启的情况，会通过接口状态表现出来，其中 up 表示链路工作正常或线路协议匹配成功，down 表示未连接到 LAN 或协议匹配失败。

（2）综合故障诊断网络日志实战

网络设备日志在进行综合故障诊断时可以提供帮助，当涉及连接故障诊断时更是如此。例如，假设某个用户抱怨应用程序不能从外部服务器下载数据。通过获取此用户机器的 IP 地址，然后找出它是在什么时候使用这个应用程序的，就可以快速搜索防火墙的日志，寻找为建立所需连接而进行的（被拒绝的）尝试。如果防火墙对所有允许的连接也进行了日志记录，而能够从日志中找到到达此远程站点的有效连接。那么，从这个事实中可以看出问题最有可能与远程服务器或应用程序有关，而不是同自己的边界防御配置有关。

7.10.1 路由器日志分析

路由器日志倾向于只包含最基本的网络信息，路由器日志在判断 ARP 病毒攻击和识别特定类型的活动（如未经授权的连接尝试和端口扫描）时极为有用。下面我们一起分析一段路由器日志。

```
    Mar 18 06:15:30 [192.168.0.10]  356118.%SEC-6-IPACCESS-LOGP:list 102 tcp
172.14.16.20(1846) ->10.20.10.18(80),1 packet
    Mar 18 06:15:30 [192.168.0.10]  356118.%SEC-6-IPACCESS-LOGP:list 102 tcp
172.14.16.20(1846) ->10.20.10.19(80),1 packet
    Mar 18 06:15:31 [192.168.0.10]  356118.%SEC-6-IPACCESS-LOGP:list 102 tcp
172.14.16.20(1846) ->10.20.10.20(80),1 packet
    ... ...
```

第一条日志主要告诉我们有人连接到 HTTP 主机，且该路由器阻塞了这条连接。然而往下却又有上千条类似这样的条目，每一个条目都涉及不同目的主机上的 TCP 端口 80，这意味着有人正在对网络进行扫描，其目的就是要寻找 Web 服务器。

7.10.2 交换机日志分析

为了诊断网络故障，有必要启用路由器、交换机的日志功能，将他们集中保存在日志服务器。下面举了几条思科、华为、中兴交换机上产生的典型日志（日志种类远不止下面介绍的这几个）。

举例 1：Cisco 3750 下联两台 2960 分别连接两台做了负载均衡的服务器，在 Cisco 3750 上的日志总是重复两条，其他日志无法查看。

日志内容：

```
    2418459: Oct 9 08:24:28: %SW_MATM-4-MACFLAP_NOTIF: Host 0201.0a80.6616 in vlan
20 is flapping between port Gi1/0/10 and port Gi1/0/9
    2418460: Oct 9 08:24:29: %SW_MATM-4-MACFLAP_NOTIF: Host 0202.0a80.6616 in vlan
20 is flapping between port Gi1/0/9 and port Gi1/0/10
    2418461: Oct 9 08:24:44: %SW_MATM-4-MACFLAP_NOTIF: Host 0202.0a80.6616 in vlan
20 is flapping between port Gi1/0/10 and port Gi1/0/9
    2418462: Oct 9 08:24:44: %SW_MATM-4-MACFLAP_NOTIF: Host 0201.0a80.6616 in vlan
20 is flapping between port Gi1/0/10 and port Gi1/0/9
    2418463: Oct 9 08:24:58: %SW_MATM-4-MACFLAP_NOTIF: Host 0202.0a80.6616 in vlan
20 is flapping between port Gi1/0/10 and port Gi1/0/9
    2418464: Oct 9 08:24:59: %SW_MATM-4-MACFLAP_NOTIF: Host 0201.0a80.6616 in vlan
20 is flapping between port Gi1/0/9 and port Gi1/0/10
```

分析：交换机交换数据包是通过 MAC 地址转发的，交换机会有一个 MAC 地址表存储每个端口对应的目的 MAC 地址，也就是收到数据包后查看目的 MAC 地址，然后转发到相应的端口，如果两个端口同时存在相同的 MAC 地址，就会出现抖动，也就是上面写的：

```
    VLAN20中的主机 0201.0a80.6616在G1/0/9和G1/0/10接口抖动
```

解决方案：清空 MAC 地址表和 ARP 地址表（待 ARP 表项超时，MAC 地址表就正常了）。

举例 2:

```
    Nov 22 22:30:08 2010 Quidway_S6500BJ SHELL/5/CMD:task:vt0 ip:192.168.0.10
user:* * command:stp
```

这条日志记录表示一个用户在 Nov 22 22:30:08 2010，从 IP 地址为 192.168.0.10 的设备登录到名为 Quidway_S6500BJ 的交换机上，执行了 stp 命令。

举例 3：

```
    Nov 23 20:30:18 2010 Quidway_S6500_BJ ARP/5/DUPIP:IP address 192.168.0.10
collision detected, sourced by 0029_d4d6_7e08 on GigabitEthernet1/0/1 of VLAN70 and
0029_d4d4_74a3 on GigabitEthernet 1/0/1 of VLAN70
```

这条日志表示在 Nov 23 20:30:18 2010，交换机 Quidway_S6500_BJ 属于 VLAN70 的千兆端口 GigabitEthernet1/0/1 下连客户机，发生了 IP 地址冲突，地址为 0029_d4d6_7e08 和 0029_d4d4_74a3。

举例 4：

```
    2010 May 12 13:15:41 %SYS-4-P2_WARN: 1/Host 00:50:fd:06:08:c0 is flapping
between port 1/2 and port 4/30
```

反复出现这种日志，典型症状是 CPU 利用率非常高，转发几乎停滞，这是因为交换机出现环路，导致广播风暴。

举例 5：

```
   An alarm 18710 level 6 occurred at 10:21:10 05/12/2010 UTC sent by MEC 1 %IP%
Interface up on vlan2
```

这是一个由 VLAN2 导致三层端口出现故障之后，在中兴交换机上产生的告警日志。

举例 6：

```
   An alarm 19716 level 6 occurred at 10:16:50 09/01/2010 UTC sent by NPC 2 %ARP%
The source IP address 208.80.160.120 conflicts with our IP address of vlan2
```

这是中兴交换机上一条 VLAN2 中 IP 地址冲突产生的告警。

举例 7：

```
   08:59:20 05/12/2010 UTC alarm 22780 occurred %MAC% [Module 70] [MAC Table] [MAC
00D0.D0C0.94B0 VLAN 2] From Port xgei_4/2 To Port gei_6/19 sent by NPC 5ge
   08:59:21 05/12/2010 UTC alarm 22780 occurred  %MAC% [Module 70] [MAC Table] [MAC
00D0.D0C0.94B0 VLAN 2] From Port xgei_4/2 To Port gei_6/19 sent by MEC 1i
```

这是一个 MAC 漂移的告警信息。

7.10.3　防火墙日志分析

防火墙是位于内外网之间的咽喉要道，所有流量都要通过防火墙，这样防火墙对每个包都要进行检查。下面分别以 Cisco 的 PIX 防火墙和 ASA 为例，分析防火墙日志。

1. PIX 日志

PIX 日志以一个百分号%开始，其后加一个格式化字符串，一条完整的日志信息可以表示为：

```
   %PIX_Level_Message_number:Message_text
```

- PIX：设备标识符，说明日志信息是 PIX 系列防火墙。
- Level：日志级别，说明该日志的严重程度，PIX 日志也是分为 8 个级别，分别从 0～7 依次递减。

接下来我们来看几个日志的例子：

（1）邮件防护日志

```
   %PIX_2_108002:SMTP replaced string : out source _address in inside_address
data:string
```

这条日志表示：当用户在防火墙上使用了 fixup protocol smtp 时，就打开了邮件防护功能，这样一来防火墙可以限制到达邮件服务器的 SMTP 消息、隐藏 SMTP 标题等，如果某个邮件

地址部分包含了非法字符，PIX 就会将非法字符用空格替代。

（2）登录验证失败日志

PIX 可以设置 ACL 限制用户对任意端口的访问，当某个用户明确禁止的访问到达 PIX，PIX 就会产生如下日志信息：

```
    %PIX_4_106023:Deny protocol src [interface name:source address/source port]dst
    Interface_name:dest_address/dest_port[type {string},code {code}] by
access_group acl_ID
```

（3）IP 碎片攻击日志

攻击者发送的 IP 数据包如果有很小的偏移或重叠碎片，那么就会绕过 IDS，所以必须用防火墙处理。举例：当一个分段数大于 12 的 IP 数据包被防火墙 Drop 时，PIX 产生如下日志信息：

```
    %PIX_4_209005:Discard IP fragment set with more than number
elements:src=IP_address,dest=IP_address,proto=protocol,id=number
```

当发现碎片重叠时，产生一条 teardrop（泪滴）系统日志：

```
    %PIX_4_106020:Deny IP teardrop fragment (size=number,offset=number) from IP
_address to IP_address
```

（4）LAND 攻击时记录的日志

LAND 攻击是一种针对 TCP 三次握手机制漏洞发起的攻击，很多系统在长时间受到这种攻击时会停止工作。PIX 防火墙检测到这种攻击时，记录日志如下：

```
    %PIX_2_106017:Deny IP due to Land Attack from IP_address to IP_address
```

2. Cisco ASA 防火墙日志

OSSIM 记录的 ASA 日志如图 7-28 所示。ASA 日志举例：

```
    <182>Dec 22 2010 14:03:05: %ASA-6-302013: Built inbound TCP connection 698572247
for outside:218.200.47.30/12026 (218.200.47.30/12026) to inside:10.1.2.97/443
(192.168.150.97/443)
```

日志格式：时间-日志编号-连接发起端-协议类型-实际源地址-实际目标地址。

“Built inbound TCP connection”建立入站 TCP 连接，代表该连接是从外部连进来。

```
    <182>Dec 22 2010 10:52:59: %ASA-6-302015: Built outbound UDP connection
697738382 for outside:117.18.82.7/123 (117.18.82.7/123) to inside:10.10.1.31/2693
(192.168.100.9/59375)Built
```

通常 Cisco 防火墙 eth0 接口定义为 outside 区，Security-Level:0，接 Router F0/0。ASA 防火墙 eth1 接口定义为 insdie 区，Security-Level:100，接 Switch 的上联口。中间的 DMZLevel 为 50。从优先级低到优先级高的方向为 inbound，反之为 outbound，所以数据从 eth1（Switch）→eth0（Router）为 outbound，从 eth0（Router）→eth1（Switch）为 inbound。

图 7-28　OSSIM 记录 Cisco ASA 日志

7.10.4　收集 CheckPoint 设备日志

本实验目的为收集 Checkpoint 设备日志，在型号 R60、R70 设备上通过测试，在 OSSIM 系统中集成了 checkpoint fw1 的插件。下面我们以 R60（fw1ngr60）为例讲解。

在使用 OSSIM 4.2 系统时，首先确定已经装好 ia32libs 包，安装方法：

```
#apt-get install lib32stdc++6 ia32-libs（安装这两个包大约100MB）。
```

再安装 fw1-loggrabber。我们可以通过 Web 界面下载，也可以直接在/usr/share/ossim/www/downloads 目录下找到 fw1-loggrabber-1.11.1-linux.tar.gz。

解压：

```
#tar zxvf fw1-loggrabber-1.11.1-linux.tar.gz
#./INSTALL.sh
install:Verzeichnis "/usr/local/fw1-loggrabber" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/bin" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/etc" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/man" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/man/man1" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/share" angelegt
install:Verzeichnis "/usr/local/fw1-loggrabber/share/fw1-loggrabber"
```

```
angelegt
   "fw1-loggrabber" - >"/usr/local/fw1-loggrabber/bin/fw1-loggraber"
   ... ...
```

当安装完成后，可以在/usr/local/目录下发现多了一个 fw1-loggrabber 目录，其中就是刚才安装的软件。然后修改/etc/profile 声明两个变量。

```
   #vi /etc/profile
   export LOGGRABBER_CONFIG_PATH=/usr/local/fw1-loggrabber/etc
   export LOGGRABBER_TEMP_PATH=/tmp
   #env（显示环境变量）
```

在/usr/local/fw1-loggrabber/etc/目录下的配置文件都是以 unix-sample 结尾的示例配置，将其分别复制成 fw1-loggrabber.conf 和 lea.conf。

下面继续配置设备日志相关信息。

将 loggrabber.conf 文件中第 42 行修改成如下设置。

```
   OUTPUT_FILE_PREFIX="/var/log/ossim/fw1-loggrabber"
```

修改/etc/ossim/agent/plugins/fw1ngr60.cfg 插件配置文件中第 15 行，如下：

```
   localtion=/var/log/ossim/fw1-loggrabber/fw1.log
```

修改 19 行，如下：

```
   create_file=true
```

保存退出，执行以下命令：

```
   #/usr/local/fw1-loggrabber/bin/fw1-loggrabber -c
/usr/local/fw1-loggrabber/etc/fw1-loggrabber.conf -l
/usr/loca/fw1-loggrabber/etc/lea.conf
```

如果提示“error while loading shared libraries:libpam.so.0:can not open shared object file:No such file or director”说明第一步没装好。

下面就可以配置 CheckPoint 防火墙了。登录到 Check Point SmartDashboard 中，选择 Manage→Servers and OPSEC Applications，在弹出的对话框中选择“New”，输入名称后选择“Client Entities”为 LEA，“Server Entities”选项为空，然后单击“Communication”按钮，然后系统会提示下载证书验证，这时可以到/var/log/ossim 目录下查看 CheckPoint 日志了。

7.10.5 Aruba（无线 AP）的日志

Aruba 的无线产品质量不错，2015 年被 HP 公司收购之后，在国内无线设备市场占有率不断攀升。下面了解一下这款 AP 在 OSSIM 中的应用实例，先看一条 Aruba AP 日志：

```
   Mar 8 18 08 36 2015 [10.0.0.252] sapd[190] <404081> <ERRS> AP sanmateo@10.0.0.127
sapd AM 00 0b 86 61 31 60 Wireless bridge detected with Transmitter 00 03 52 e7 c5 20 Rece
```

```
iver 00 03 52 e7 90 30 Channel 4 and RSSI 12
```

这条日志在 OSSIM 日志管理系统中显示如图 7-29 所示的内容。

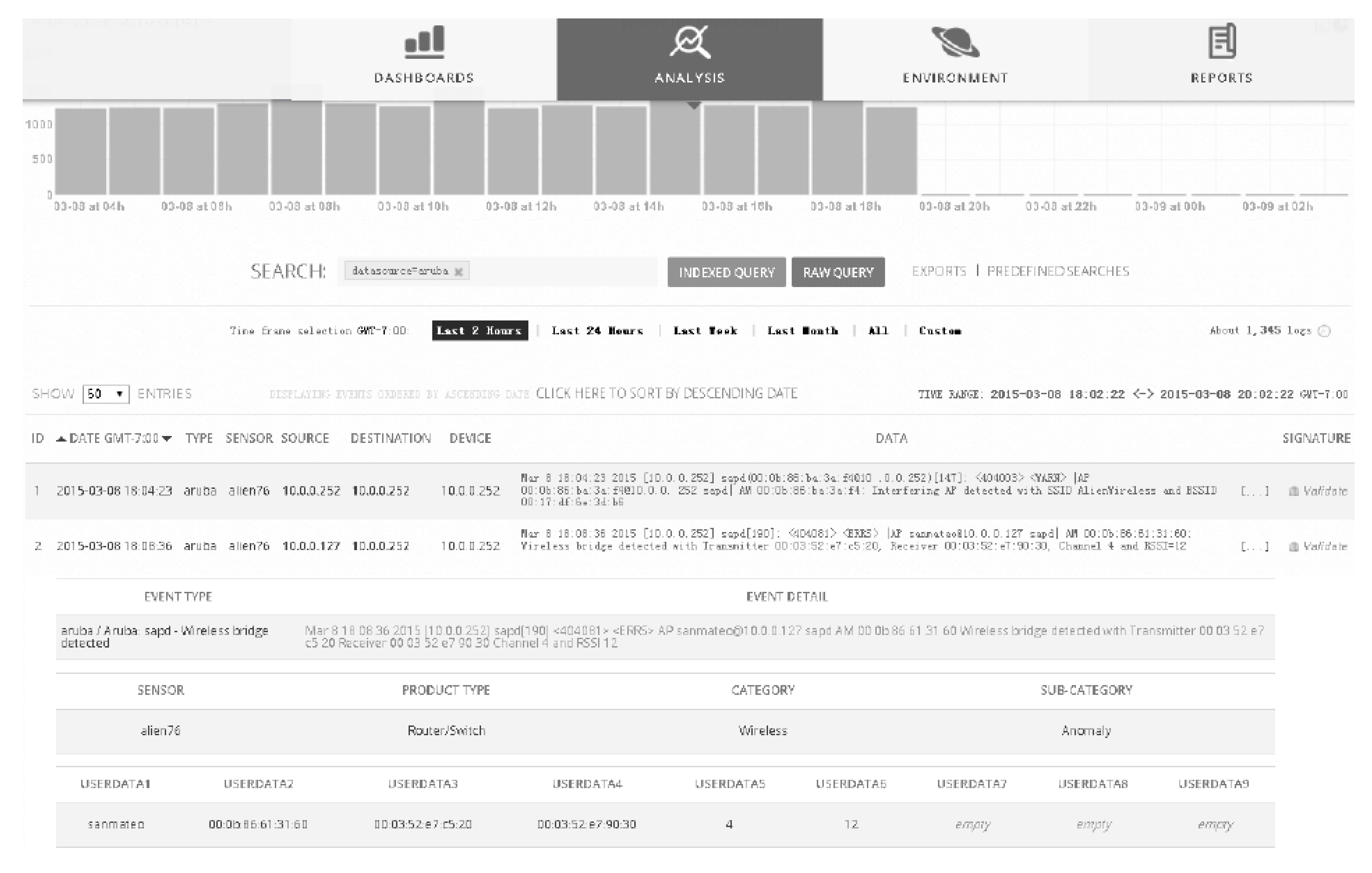


图 7-29　Aruba AP 日志

7.11　Apache 日志分析

我们先以最为常见的 Apache 服务器为例进行日志分析。Apache 服务器的日志文件中包含着大量有用的信息，这些信息经过分析和深入挖掘之后，能够最大限度地在系统管理人员及安全取证人员的工作中发挥重要作用。

7.11.1　日志作用

Apache 访问日志在实际工作中非常有用，比较典型的例子是进行网站流量统计，查看用户访问时间、地理位置分布、页面点击率等。Apache 的访问日志具有如下 4 个方面的作用：

- 记录访问服务器的远程主机 IP 地址，从而可以得知浏览者来自何处；
- 记录浏览者访问的 Web 资源，可以了解网站中的哪些部分最受欢迎；
- 记录浏览者使用的浏览器，可以根据大多数浏览者使用的浏览器对站点进行优化；
- 记录浏览者的访问时间。

7.11.2 日志格式分析

RAW log 日志，又称为原始日志。分析这种日志的基础是了解日志中每段信息表达的含义。在 Apache 中访问日志功能由 mod_log_config 模块提供，它用默认的 CLF（common log format）来记录访问日志。例如 LogFormat "%h%1%u%t %r"。下面是一条 Apache 服务器记录的实际访问日志，各列（用颜色代表对应列的含义，读者可以参看实际操作界面）含义如表 7-4 所示。

200.202.39.131 - - [21/Nov/2012:10:45:13+0800] "GET/original/warn.pngHTTP/1.1" 200 1961

远程主机 IP 地址　请求时间　时区　方法　资源 URL　协议　返回状态　发送字节

表 7-4　Apache 访问日志分析

域	内容	含义
$1	200.202.39.131	远程主机 IP 地址，%h
$2	-	占位符，%1
$3	-	占位符，%u
$4	21/Nov/2012:10:45:13	服务器完成请求处理时间，[日/月/年:小时：分钟：秒：时区]%t
$5	+0800	时区
$6	GET	方法（GET、POST）%r\
$7	/original/warn.png	资源 URL
$8	HTTP/1.1	协议
$9	200	返回状态 %s
$10	1961	发送给客户端总字节数，%b

主机地址（表 7-4 中为 200.202.39.131）后面紧跟的两项内容现在很少使用，所以用两个“-”占位符替代。

7.11.3 日志统计举例

可以用 tail 命令来实时查看日志文件的变化，但是各种应用系统中的日志会非常复杂，一堆长度超过浏览极限的日志出现在眼前时，会觉得非常无奈，怎么办呢？这时可以用 grep、sed、awk 和 sort 等筛选工具帮助你解决这个问题。下面总结了几个常见分析方法。

（1）查看 IP（$1 代表 IP）。

```
#cat access_log | awk '{print $1}'
```

（2）对 IP 排序。

```
#cat access_log | awk '{print $1}'|sort
```

（3）打印每一重复行出现的次数，“uniq -c”表示标记出重复数量。

```
#cat access_log | awk '{print $1}'|sort|uniq -c
```

（4）排序并统计行数。

```
#cat access_log | awk '{print $1}'|sort|uniq -c|sort -rn|wc -l
```

（5）显示访问前 10 位的 IP 地址，便于查找攻击源。

```
#cat access_log|awk '{print $1}'|sort|uniq -c|sort -nr|head -10
```

awk '{print $1'}表示取日志的第一段，如果换成别的日志，其 IP 地址在第 3 段，那么就要改变相应的数值。

（6）显示指定时间以后的日志（$4 代表时间）。

```
#cat access_log |awk '$4>="[23/Jul/2012:01:00:01"' access_log
```

推荐大家在排错时，同时打开多个终端，比如在一个窗口中显示错误日志，在另一个窗口中显示访问日志，这样就能够随时获知网站上发生的情况。

（7）找出访问量最大的 IP，并封掉（对排错很有帮助）。

```
#cat access_log |awk '{print $1}'|sort|uniq -c |sort -nr |more
9999 192.168.150.179
11  192.168.150.1
#iptables -I INPUT -s 192.168.150.179 -j DROP
#iptables -I INPUT -s 192.168.150.0/24 -j DROP
```

如果将上面的 Shell 做以下变形就可以得出访问量 TOP 10。

```
#cat access_log |awk '{print $1}'|sort|uniq -c |sort -nr |head -10
```

（8）找出 Apache 日志中，下载最多的几个 exe 文件（下载类网站常用，这里以.exe 扩展名举例）。

```
[root@localhost httpd]# cat access_log |awk '($7 ~/.exe/){print $10 "" $1 ""
$4""$7}' |sort -n |uniq -c |sort -nr |head -10
2 - 192.168.150.1[25/Jul/2012:05:46:05/test.exe
1 - 192.168.150.152[25/Jul/2012:05:46:47/test.exe
```

使用如下命令：

```
#cat access_log |awk `($10 >10000000 && $7 ~/.exe/) {print $7}` |sort -n|uniq
-c|sort -nr|head -10
```

这条命令经过增加一个“>10000000”的条件判断内容就可以显示出大于 10MB 的 exe 文件，并统计对应文件发生次数，这条命令对于网站日常分析是非常有帮助的，大家可以灵活使用。

7.11.4　错误日志分析

错误日志记录了服务器运行期间遇到的各种故障，以及一些普通的诊断信息，比如服务器

启动/关闭时间。错误日志和访问日志一样也是 Apache 的标准日志。它在 http.conf 配置文件中 ErrorLog logs/ error_log 处定义路径和格式，错误日志和访问日志一般放在同一个目录。

日志文件记录信息级别的高低，控制日志文件记录信息的数量和类型，都是通过 LogLevel 指令实现的，该指令默认设置的级别是 error，有关该指令中允许设置的各种选项的完整清单，请参见 http://wiki.apache.org/httpd/FAQ 的 Apache 文档。最常见的错误日志文件有两类，一类是文档错误信息，另一类是 CGI 错误信息。

（1）文档错误

文档错误和服务器应答中的 400 系列代码对应，最常见的就是 404 错误——Document Not Found（文档没有找到）。这种错误在用户请求的 URL 不存在的时候出现，一般是由于用户输入的 URL 错误，或者由于 Web 服务器上已存在的文件被删除或移动。

错误日志中出现的记录如下所示：

```
[Sun Dec 23 06:17:18 2012][error] [client 192.168.150.16] File does not
exist:/usr/local/apache2/docs/index.html
[Sun Dec 23 06:27:58 2012] [notice] Apache/2.2.16(Debian)PHP/5.3.3-7+squeeze14 with
Suhosin-Patch proxy_html/3.0.1 mod_ssl/2.2.16 OpenSSL/0.9.80 mod_perl/2.0.4 Perl/v5.10.1
configured --resuming normal operations
[Sun Dec 23 07:27:18 2012] [error] [client 192.168.150.149 PHP Warning: preg_replace():
Compilation failed:missing terminating] for character class at offset 9 in
/usr/share/ocsinventory-server/ocsreports/groups.php on line
126,referer:https://192.168.150.28/ossim/ocsreports/index.php?multi=37
[Sun Dec 23 09:17:28 2012] [warn] RSA server certificate CommonName (CN) `localhost`
does NOT match Server name!?
```

错误日志和访问日志格式类似，不同之处在[error]这一项，它表示记录级别，为方便开发和调试分为 0~7 级，含义同表 7-4 所示内容。日志中的错误级别由配置文件中 LogLevel 指令负责调整，它的主要作用是控制错误日志的详细程度（在 httpd.conf 配置文件中说明）。

记录级别从 0～7 级，日志记录量是从小向大方向增长，7 级为最高，这和思科等路由器的 debug 模式相同。各级别的关系如图 7-30 所示。

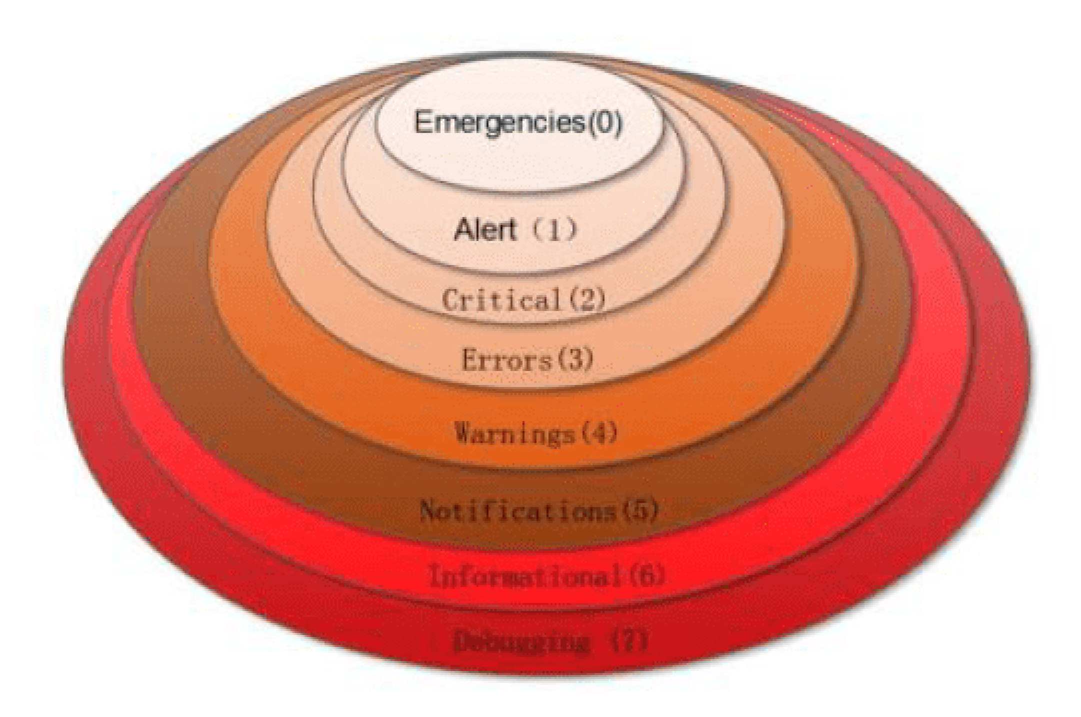


图 7-30　日志记录等级

从图 7-30 可以看出，日志按严重程度分为 8 组，从高到低依次为紧急（0 级别）、报警（1 级）、关键、错误、一般错误、警告、通知及消息调试。圆圈越大，则说明所记录的日志信息量越多，8 级（Debugging，调试级）包含了上面 7 级记录的所有信息，所以它的日志量最大，因此不要在工作的设备上启用这一级。

在正常运行的服务器上，一般有两种错误信息，一种是文档错误，最常见的就是 400 系列错误，例如文件被移走或删除而出现的 404 错误等；另一种是 CGI 错误，主要由 CGI 程序引起。

（2）CGI 错误

错误日志还能诊断异常行为的 CGI 程序。为了进一步分析和处理方便，CGI 程序输出到 STDERR（Standard Error，标准错误设备）的所有内容都将直接进入错误日志。如果 CGI 程序出现了问题，错误日志就会告诉我们有关问题的详细信息。

下面是一个例子，它是调试 CGI 代码时，错误日志中出现的一个错误记录：

```
[Sat Feb 05 14:01:29 2012] [error] [client 220.106.0.18] Premature end of script headers:
/usr/local/apache2/cgi-bin/doc/index.cgi Global symbol "$rv" requires explicit package name at
/usr/local/apache2/cgi-bin/doc/index.cgi line 70.
[Sat Feb 05 14:08:24 2012] [error] [client 220.106.0.18] Premature end of script headers: cgi
[Sat Feb 05 14:01:52 2012] [error] [client 220.106.0.18] (Exec format error: exec of '/
usr/local/apache2/cgi-bin/doc /Search/cgi-bin/cgi' failed
```

错误日志记录通常以行为单位。在上面给出的情况中，CGI 错误就会出现多行情况，从这一点看，Apache 日志级别的定义也不是很严格，例如在单个文件记录所有日志时，无论使用哪种错误级别，在日志中总会显示 Notice 级别的信息，这些信息虽然是提醒程序员需要注意，有时却显得多余。所以建议大家使用 Rsyslog 记录日志，这样就不会出现上述问题。

默认将错误文件放在 Apache 配置文件中 ServerRoot 的 logs 目录下，一般路径位于 /var/log/apache2/error_log，这里假设文件名为 error_log（有的系统为 error.log）。

如果在配置文件中停止输出错误日志，例如：

```
errorlog /dev/null
```

一旦服务器崩溃就会丢失很多有价值的调试信息，所以在万不得已的情况下不要使用此方法，不过有时错误日志会变得非常大。

7.12 Nginx 日志分析

Nginx 日志和 Apache 日志相似，同样分为访问日志和错误日志，它们都在 nginx.conf 配置文件中定义。为了分析 Nginix 日志读者需要了解其格式字段的含义。

7.12.1 基本格式

以 Nginx 默认的日志格式为例其格式如下：

```
$remote_addr - $remote_user [$time_local] "$request" $status $body_bytes_sent
"$http_referer" "$http_user_agent"
```

各字段的含义分别为：

- $remote_addr：请求者 IP。
- $remote_user：HTTP 授权用户，如果不使用 Http-based 认证方式，其值为空。
- [$time_local]：服务器时间戳。
- "$request"：HTTP 请求类型（如 GET，POST 等）+HTTP 请求路径（不含参数）+HTTP 协议版本。
- $status：服务器返回的状态码（如 200，404，5xx 等）。
- $body_bytes_sent：服务器响应报文大小，单位 byte。
- "$http_referer"：referer 字段值。
- "$http_user_agent"：User Agent 字段。

了解日志基本格式对于以后我们分析日志有好处，下面是为 /var/log/nginx/访问日志的例子，如图 7-31 所示。

```
192.168.91.1 - - [06/Feb/2015:02:33:37 -0500] "GET / HTTP/1.1" 200 157 "-" "Mozi
lla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
```

图 7-31　一段 nginx 日志实例

以下列举常用的日志分析命令，图 7-32 所示命令是根据状态码进行请求次数排序：

```
alienvault:/var/log/nginx# cat localhost.access.log |cut -d '"' -f3 |cut -d ' '
-f2 |sort |uniq -c |sort -r
      2 404
      1 200
alienvault:/var/log/nginx# _
```

图 7-32　根据状态码排序

使用 awk 命令也可以实现以上功能，操作效果如图 7-33 所示。

```
alienvault:/var/log/nginx# awk '{print $9}' localhost.access.log |sort |uniq -c
|sort -r
      2 404
      1 200
alienvault:/var/log/nginx# _
```

图 7-33　用 awk 实现排序

接着，查看 404 请求，如何找到这些 URL，操作效果如图 7-34 所示。

```
 |sort -r
      2 192.168.91.1 - - [06/Feb/2015:02:33:37 -0500] "GET /favicon.ico HTTP/1.1
" 404 143 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) li
ke Gecko"
alienvault:/var/log/nginx# _
```

图 7-34　查找 URL

接下来考虑，包含 favicon.ico（也可以是一个网址的一部分），找到这些请求的 IP 地址，使用 awk 命令，操作效果如图 7-35 所示。

```
alienvault:/var/log/nginx# awk -F \" '($2 ~"/favicon.ico"){print $1}' localhost.
access.log |awk '{print $1}' |sort |uniq -c |sort -r
      3 192.168.91.1
alienvault:/var/log/nginx#
```

图 7-35　查找 IP

7.12.2　将 Nginx 日志发送到 Syslog

在 Nginx Plus 中直接支持将 access_log 和 error_log 发送到 syslog，但在免费的版本 Nginx 中，默认情况下日志不支持 syslog 功能，所以只能找第三方模块，支持 syslog 的 Nginx 第三方模块，下载地址为 https://github.com/yaoweibin/nginx_syslog_patch，具体安装/配置方法可以直接看下载地址中的 README 文件。

7.13　FTP 日志分析

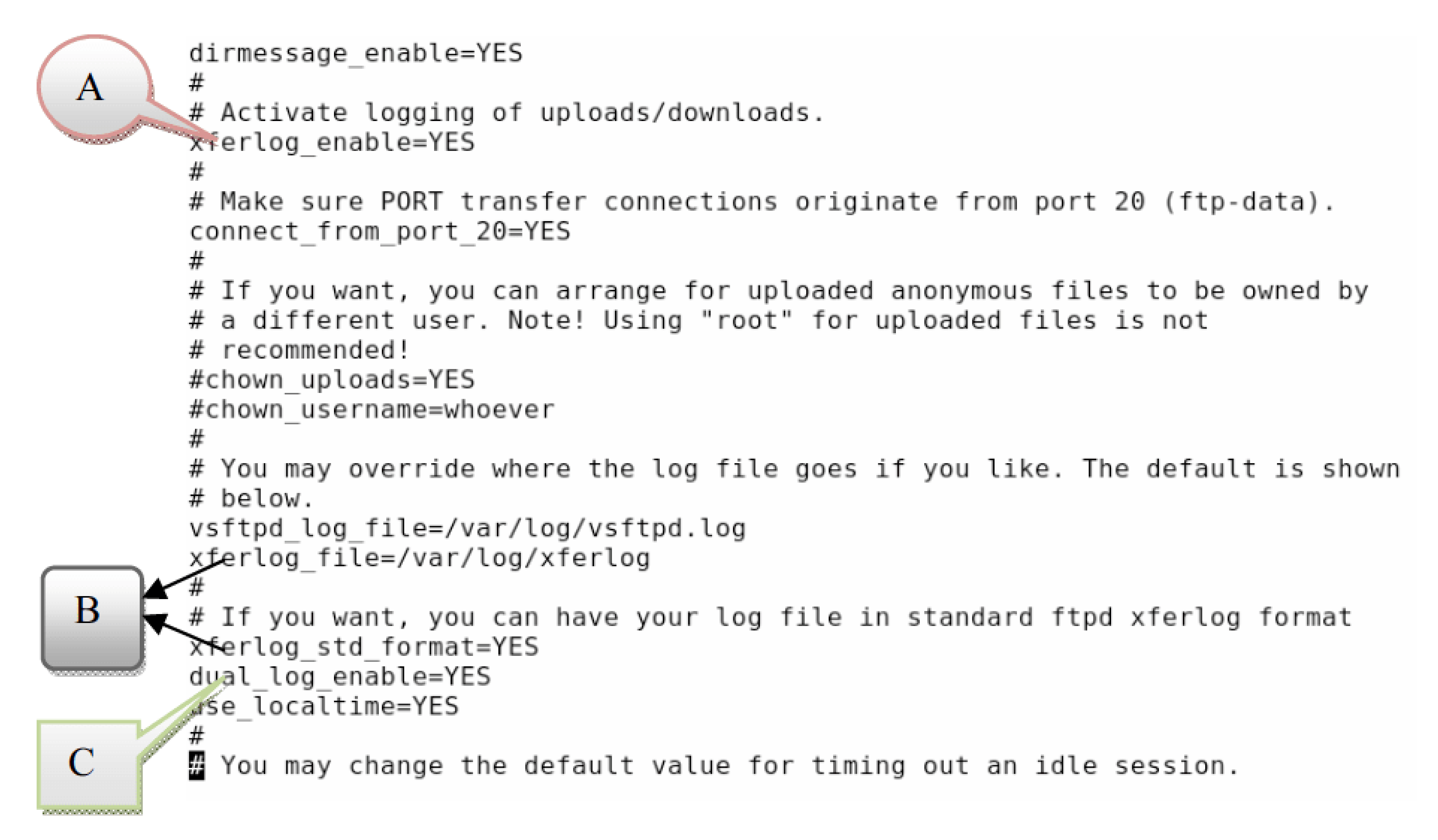
FTP 是老牌的文件传输协议，在网络中应用非常广泛。本节对 Vsftp 服务器的日志进行重点讨论，在 Redhat Linux 系统下 Vsftp 的配置文件在/etc/vsftp/vsftp.conf 文件中。默认情况下，Vsftp 不单独记录日志，也就是说不会输出到一个单独的文件中存储，而是统一存放到/var/log/messages 文件中。Vsftp 日志实例显示如下。

```
[root@localhost httpd]# cat /var/log/messages |grep vsftp
Jan  4 01:03:52 localhost vsftpd: Fri Jan  4 06:03:52 2013 [pid 9298] CONNECT: Client "192.168.15
0.1"
Jan  4 01:03:57 localhost vsftpd: Fri Jan  4 06:03:57 2013 [pid 9297] [tet] FAIL LOGIN: Client "1
92.168.150.1"
Jan  4 01:14:01 localhost vsftpd: Fri Jan  4 06:14:01 2013 [pid 9392] CONNECT: Client "192.168.15
0.1"
Jan  4 01:14:03 localhost vsftpd: Fri Jan  4 06:14:03 2013 [pid 9391] [test] OK LOGIN: Client "19
2.168.150.1"
Jan  4 01:16:02 localhost vsftpd: Fri Jan  4 01:16:02 2013 [pid 9423] CONNECT: Client "192.168.15
0.1"
Jan  4 01:16:04 localhost vsftpd: Fri Jan  4 01:16:04 2013 [pid 9422] [test] OK LOGIN: Client "19
2.168.150.1"
```

通过在 messages 中过滤的方法可以看到 Vsftp 的客户机连接日志，但这段日志只反映了少量信息，如果需要查看更详细的信息该如何操作？下面来看看怎样配置/etc/vsftp/vsftp.conf 文件。

7.13.1　FTP 日志分析

如何将 Vsftp 服务器的日志单独输出到某个文件下呢？我们需要在 vsftp.conf 配置文件中修改三处，如下所示。



```
dirmessage_enable=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to be owned by
# a different user. Note! Using "root" for uploaded files is not
# recommended!
#chown_uploads=YES
#chown_username=whoever
#
# You may override where the log file goes if you like. The default is shown
# below.
vsftpd_log_file=/var/log/vsftpd.log
xferlog_file=/var/log/xferlog
#
# If you want, you can have your log file in standard ftpd xferlog format
xferlog_std_format=YES
dual_log_enable=YES
use_localtime=YES
#
# You may change the default value for timing out an idle session.
```


在上面三处配置中，我们做了修改：

（1）标识 A：启用 xferlog_enable=YES，它表示把客户机登录服务器后上传或下载的文件具体信息记录下来。

（2）标识 B：启用 xferlog_file=/var/log/xfer.log，它表示把上传下载记录写到指定文件，也就是/var/log/xferlog 文件。

（3）标识 C：启用 dual_log_enable=YES，它表示启用双份日志，一份日志由 xferlog 记录，同时 vsftpd.log 也记录另一份日志，注意两份日志并非互为备份，其内容不同，但各有千秋。

接下来还得解释一下/usr/bin/xferstats，它是日志统计工具，用于计算传输了多少文件并创建日志文件。注意：在 Linux 系统中一定要安装 xferstats 的包后，才能对它进行操作。

7.13.2 分析 vsftpd.log 和 xferlog

Vsftpd.log 和 xferlog 是 Vsftp 服务器记录日志的来源，下面重点对这两种日志文件的格式进行分析。

（1）vsftpd.log 实例分析

首先打开 vsftpd.log.1 文件，查看它的日志结构。

```
[root@localhost log]# cat vsftpd.log.1
Thu Jan  3 14:06:07 2013 [pid 3635] CONNECT: Client "192.168.150.1"
Thu Jan  3 14:06:10 2013 [pid 3634] [test] OK LOGIN: Client "192.168.150.1"
Thu Sep  5 04:08:19 2013 [pid 4325] CONNECT: Client "192.168.150.1"
Thu Sep  5 04:08:21 2013 [pid 4324] [test] OK LOGIN: Client "192.168.150.1"
Thu Sep  5 04:08:58 2013 [pid 4451] CONNECT: Client "192.168.150.28"
Thu Sep  5 04:09:01 2013 [pid 4450] [test] OK LOGIN: Client "192.168.150.28"
Thu Sep  5 04:12:39 2013 [pid 4467] CONNECT: Client "192.168.150.147"
Thu Sep  5 04:12:41 2013 [pid 4466] [test] OK LOGIN: Client "192.168.150.147"
Thu Sep  5 04:12:54 2013 [pid 4468] [test] OK MKDIR: Client "192.168.150.147", "/home/test/huge"
```

上面日志仅反映了部分 FTP 登录情况，例如登录 IP 地址、用户名。但下载软件内容不会记录下来，有时网管恰好关心这一段日志信息，这时我们需要同时参考 xferlog 日志，还记得上面说过的 xferstats 工具吗？

（2）Xferlog 日志实例分析

xferlog 日志会记录 FTP 会话详细信息，它能够显示客户机向 FTP Server 上传/下载的文件路径、名称及认证方式等信息，下面我们查看该文件的具体内容。

```
[root@localhost log]# cat xferlog.1
Thu Jan  3 14:24:48 2013 1 127.0.0.1 8646 /home/test/nmbd.log b _ i r test ftp 0 * c
Thu Jan  3 14:36:46 2013 60 192.168.150.1 0 /home/test/syslog.jpg a _ i r test ftp 0 * i
Thu Jan  3 15:06:58 2013 1 192.168.150.151 8646 /home/test/nmbd.log b _ o r test ftp 0 * c
Thu Jan  3 15:07:01 2013 1 192.168.150.151 0 /home/test/123.txt b _ o r test ftp 0 * c
Thu Sep  5 04:03:51 2013 1 192.168.150.206 0 /home/test/syslog.jpg b _ o r test ftp 0 * c
```

Xferlog 日志格式如下，其解析见表 7-5。

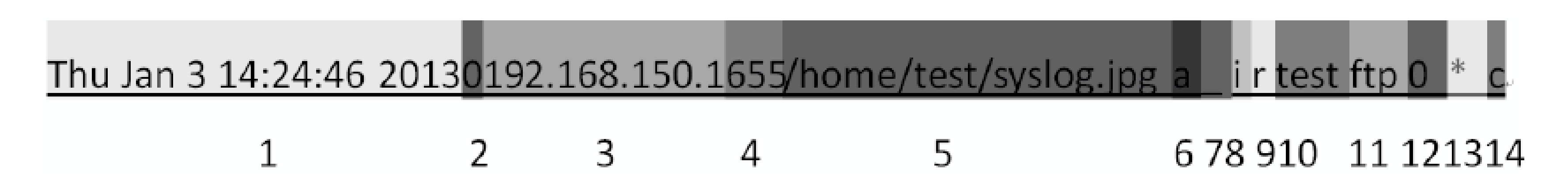


表 7-5　xferlog 日志格式

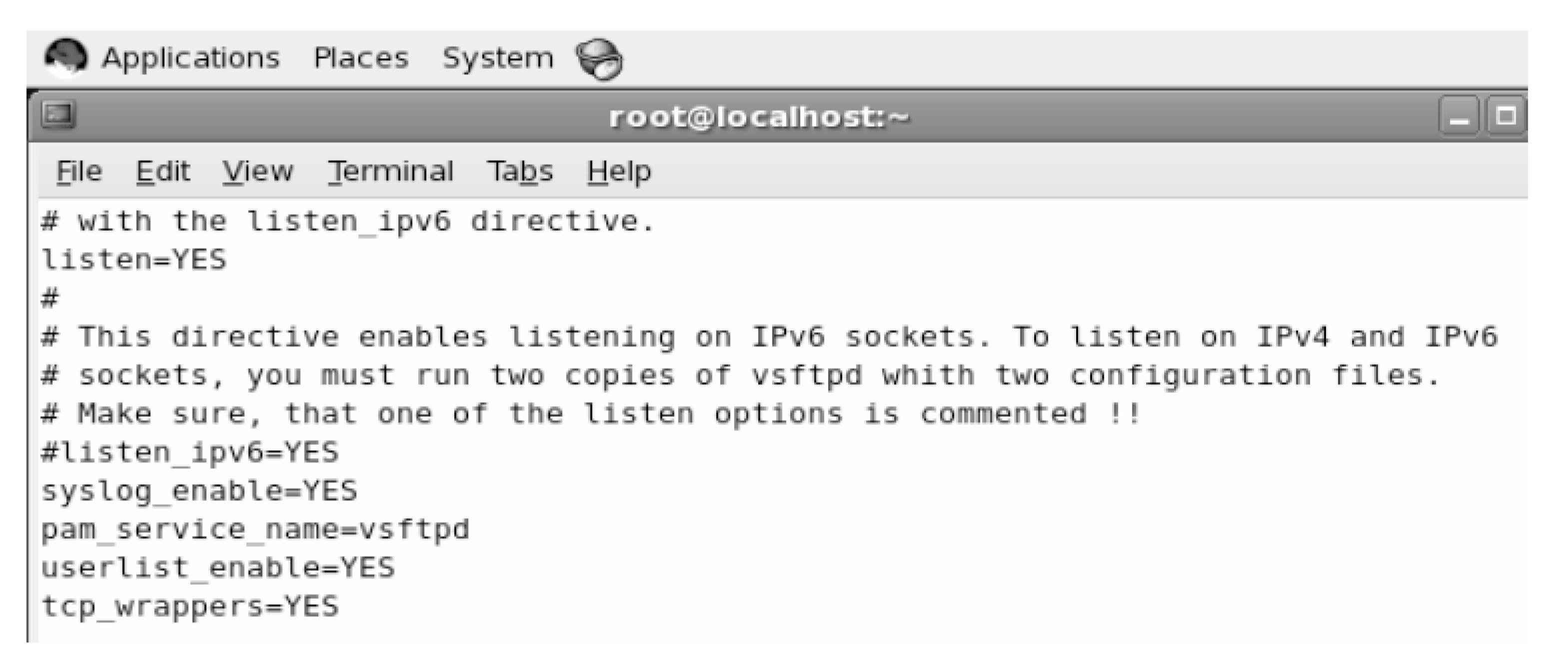
域	内容	含义
1	Thu Jan 3 14:24:46 2013	访问时间
2	0	传输文件所用的时间
3	192.168.150.1	远程主机名或 IP
4	655	文件大小，单位 byte
5	/home/test/syslog.jpg	文件路径
6	a	传输类型： a：表示 ASCII 传输，用于文本类型； b：表示二进制传输，用于程序、多媒体文件
7	_	特殊处理标志： _：不做处理 C：压缩格式 U：非压缩格式 T：Tar 文件格式
8	i	文件传输方向： o：从 FTP 服务器向客户端传输 i：从客户端向 FTP 服务器传输
9	r	访问模式： a：匿名用户 g：来宾（guest）用户 r：真实用户
10	test	用户名
11	ftp	FTP 服务器名称，通常为 ftp
12	0	认证方式，一般用 0 表示
13	*	认证用户 ID，在无须认证时用*表示，如果 vsftpd 使用了 PAM 配置，这里会有虚拟用户名显示
14	c	传输状态：c 表示完成，i 表示传输异常

这里的认证是结合 PAM（一种可插入的安全验证模块）的方式，主要是为了保证安全。在企业中常会用到 Vsftp+Pam+Postgresql 的架构，在这种架构中我们可以设置为用 MD5 工具来验证密码，这样客户机必须用 MD5 加密的密码登录系统才能成功获取文件。

7.13.3 将 Linux 的 Vsftp 日志发送到 OSSIM

本节给大家介绍如何将 RHEL5 Linux 下的 Vsftp 服务器日志发送到 OSSIM 4.8 系统，步骤如下：

（1）在 vsftpd.conf 中添加 Syslog_enable=YES 一行，如图 7-36 所示。



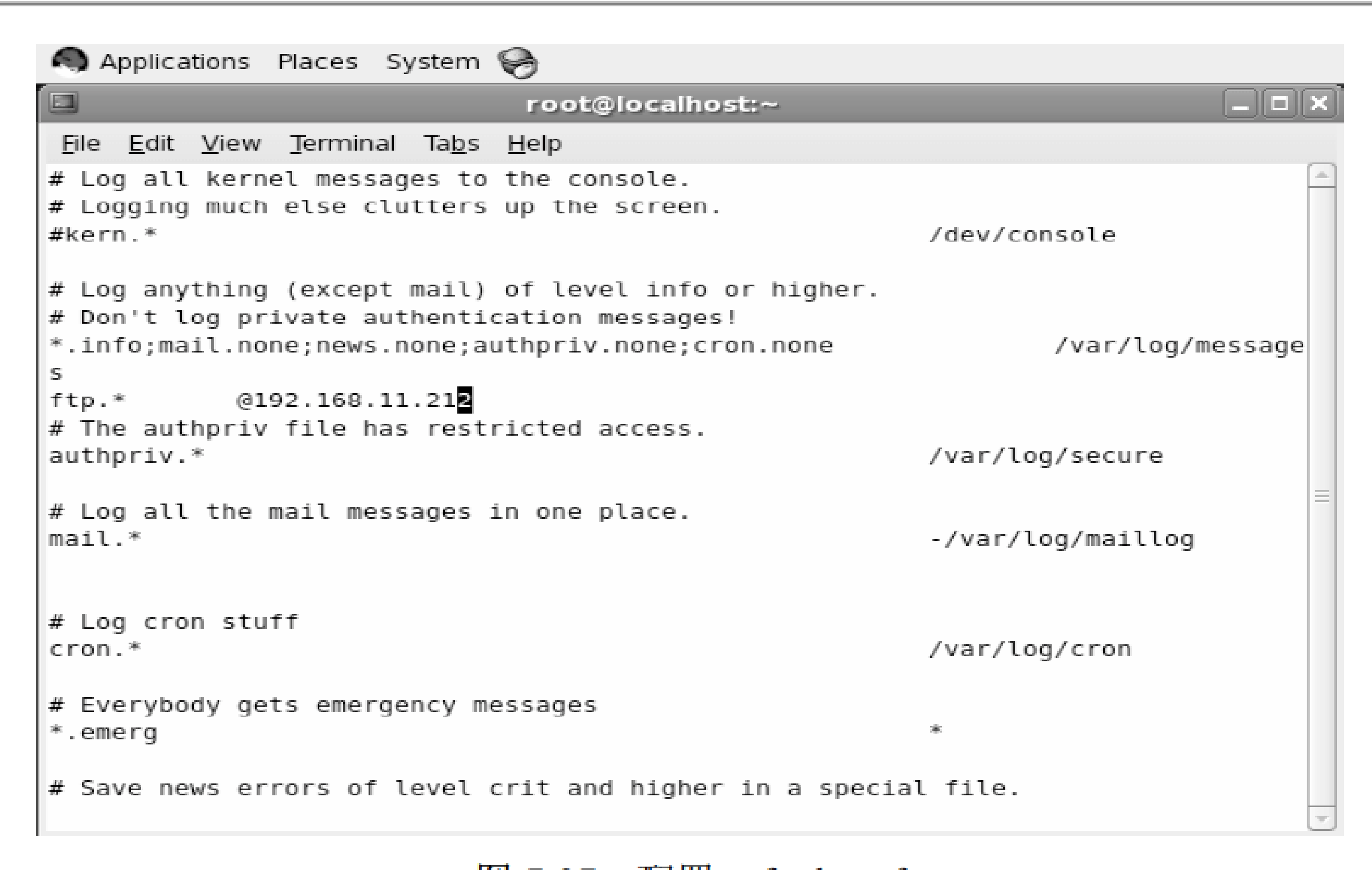
```
# with the listen_ipv6 directive.
listen=YES
#
# This directive enables listening on IPv6 sockets. To listen on IPv4 and IPv6
# sockets, you must run two copies of vsftpd whith two configuration files.
# Make sure, that one of the listen options is commented !!
#listen_ipv6=YES
syslog_enable=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

图 7-36　配置 vsftpd.conf

（2）在 syslog.conf 或者 rsyslog.conf 中添加一行，如图 7-37 所示。

```
ftp.*    @192.168.11.212
```

这里“192.168.11.212”为 OSSIM　SensorIP 地址。



```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                  /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none               /var/log/message
s
ftp.*       @192.168.11.212
# The authpriv file has restricted access.
authpriv.*                                               /var/log/secure

# Log all the mail messages in one place.
mail.*                                                   -/var/log/maillog


# Log cron stuff
cron.*                                                   /var/log/cron

# Everybody gets emergency messages
*.emerg                                                  *

# Save news errors of level crit and higher in a special file.
```

图 7-37　配置 vsftpd.conf

（3）在 CONFIGURATION → COMPONENTS → ALIENVAULT CENTER → SENSOR CONFIGURATION 下的 COLLECTION 中添加 vsftpd 插件并启用，效果如图 7-38、图 7-39 所示。

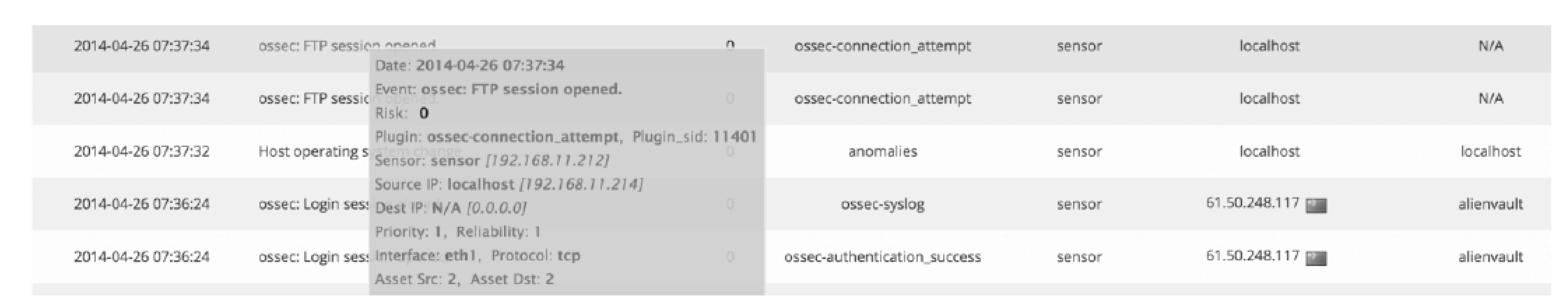


图 7-38　vsftpd 插件启用效果

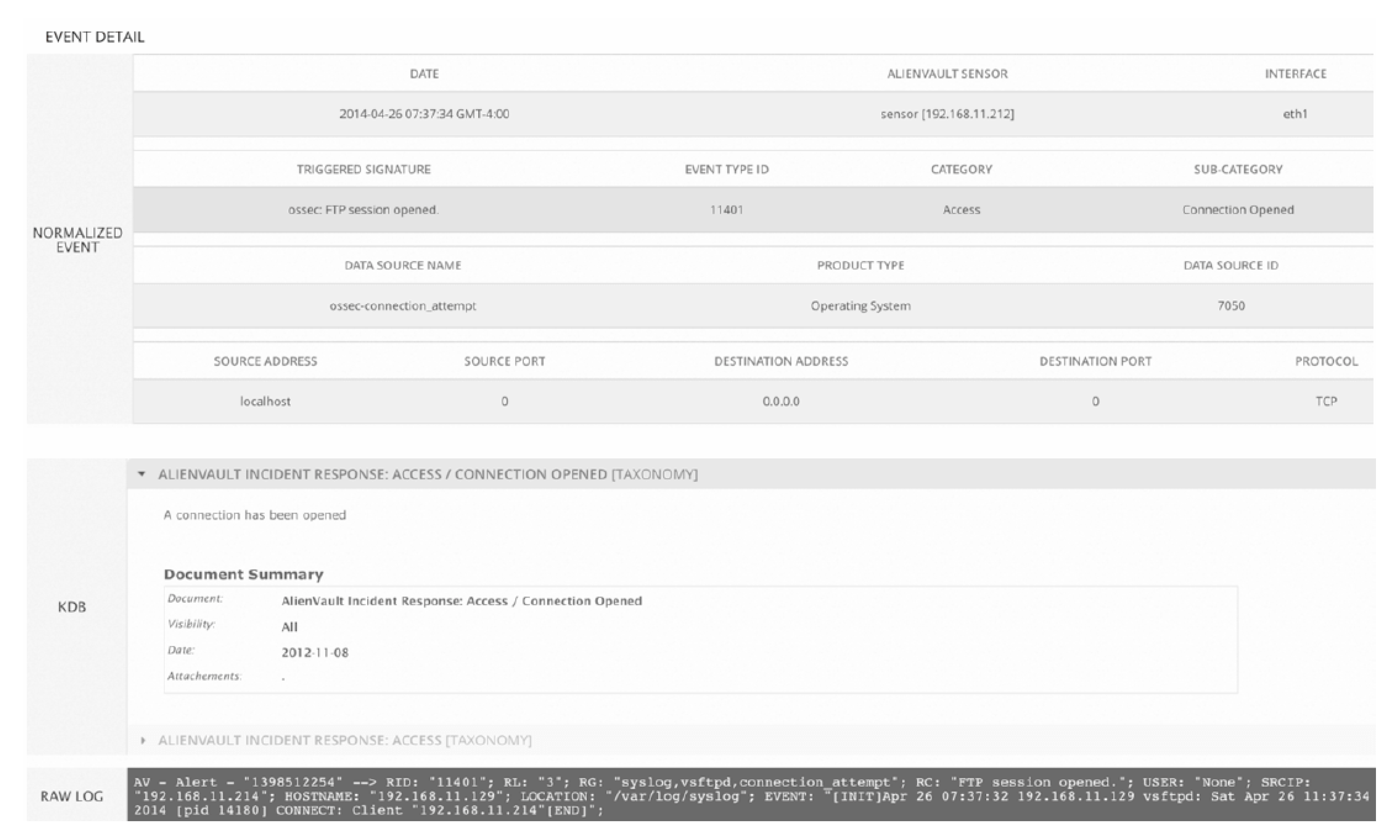


图 7-39　vsftpd 插件启用效果

Raw Log 日志会记录到/var/ossec/logs/alerts/2014/apr 目录下，例如：

```
/var/ossec/logs/alerts/2014/apr/ossec-alerts-26.log
```

每天产生一个文件，多次链接分条目，记录在一个文件中。

7.14　iptables 日志分析

7.14.1　iptables 日志分析

防火墙除了能进行有效控制网络访问之外，还有一个很重要的功能就是能清晰地记录网络

上的访问，并自动生成日志进行保存。无论是后面我们谈到的 PIX、ASA 或是 CheckPoint 防火墙，它们产生的日志内容均类似。任何连接或者请求，例如 TCP、UDP、ICMP 连接记录、连接的流量信息、连接建立时间等，防火墙日志都会将其逐一体现。所以归纳起来，防火墙日志大致包含消息发送源 IP 地址、消息目的 IP、消息流向、消息的内容，以及应用几方面。

防火墙每天要产生大量的日志文件，防火墙管理员针对这未经任何处理和分析的庞大的日志进行管理是很困难的。因此，日志的统计和分析现在已经成为防火墙功能中必不可少的一项，管理员不但可以按照不同的需求来查找日志、审计日志，还可以分析网络带宽的利用率、各种网络协议和端口的使用情况等。防火墙日志还会产生安全警告以及一些对网络安全管理很有帮助的信息。这极大地方便了管理员对防火墙的安全管控。

本节以 Linux 下的 iptables 为例讲解防火墙日志。下面是一段通过“-j LOG”方式获取的 iptables 一行日志内容：

```
   Jun 19 17：20:04 webkernel： NEW DRAP
IN=eth0OUT=MAC=00:10:4b:cd:7b:b4:00:e0:le:b9:04：al:08:00 SRC=192.168.150.1
DST=192.168.150.152 LEN=20 TOS=0X00 PREC=0x00 TTL=249 ID=10492 DF PROTO=UDP SPT=53
DPT=32926 LEN=231
```

对于此日志解释见表 7-6 所示。

表 7-6　Iptables Log 字段解释

序号	字段名称	含义
1	Jun 19 17:20:24	日期时间，由 syslog 生成
2	Web	主机名称
3	Kernel	进程名由 syslogd 生成 kernel 为内核产生的日志说明 netfilter 在内核中运行
4	NEW_DRAP	记录前缀，由用户指定—log-prefix “NEW_DRAP”
5	IN=eth0	数据包进入的接口，若为空表示本机产生，接口还有 eth0、br0 等
6	OUT=	数据包离开的接口，若为空表示本机接收
7	MAC=00:10:4b:cd:7b:b4:00:e0:le:b9:04:al	00:10:4b:cd:7b:b4　为目标 MAC 地址 00:e0:le:b9:04:al　为源 MAC 地址
8	08:00	08:00　为上层协议代码，即表示 IP 协议
9	SRC=192.168.150.1	192.168.150.1 为源 IP 地址
10	DST=192.168.150.152	192.168.150.152 为目标 IP 地址
11	LEN=20	IP 封包+承载数据的总长度（MTU）
12	TOS=0x00	IP 包头内的服务类型字段，能反映服务质量包括延迟、可靠性和拥塞等
13	PREC=0x00	服务类型的优先级字段
14	TTL=249	IP 数据包的生存时间
15	ID=10492	IP 数据包标识
16	DF	DF 表示不分段，此字段还可能为 MF/FRAG

（续表）

序号	字段名称	含义
17	PROTO=UDP	传输层协议类型，它代表上层协议是什么，可分为 TCP、UDP、ICMP 等
18	SPT=53	表示源端口号
19	DPT=32926	表示目的端口号
20	LEN=231	传输层协议头长度
21	SEQ= 内容略	TCP 序列号
22	ACK=内容略	TCP 应答号
23	WINDOWS=内容略	IP 包头内的窗口大小
24	RES	保留值
25	CWR/ECE/URG/ACK/PSH/RST/SYN/FIN	TCP 标志位
26	URGP=	紧急指针起点
27	OPT（内容略）	IP 或 TCP 选项，括号内为十六进制
28	INCOMPLETE[65535 bytes]	不完整的数据包
29	TYPE=CODE=ID=SEQ=PARAMETER=	当协议为 ICMP 时出现
30	SPI=0xF1234567	当前协议为 AHESP 时出现
31	SYN	TCP-Flags 中的 SYN 标志，此外还有 FIN/ACK/RST/URG/PSH 几种
32	[ ]	中括号出现在两个地方，在 ICMP 协议中作为协议头递归使用；在数据包长度出现非法时用于指出数据实际长度

TOS 的功能是对不同类型的应用协议数据包打标记，这样一来当路由器看到带有这些标记的数据包后，即可通过策略路由将不同的应用协议流量转发到不同的出口链路。当然路由器设备必须支持 TOS 机制才行。

7.14.2　iptables 日志管理范例

本节介绍的内容主要是系统单独生成 iptables 的独立日志，并且按每天记录，生成 iptables 滚动日志。

（1）配置 syslogd 的配置文件/etc/syslog.conf

在文件 syslog.conf 里添加如下内容：

```
# iptables 日志
kern.warn /var/log/iptables
```

（2）使用 iptables 滚动日志

先查看并确定 logrotate 的配置文件/etc/logrotate.conf，内容如下：

```
# see "man logrotate" for details
# rotate log files weekly weekly
# keep 4 weeks worth of backlogs rotate 4
# create new (empty) log files after rotating old ones create
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory include
/etc/logrotate.d
# no packages own wtmp -- we'll rotate them here /var/log/wtmp { monthly create
0664 root utmp rotate 1 }
# system-specific logs may be also be configured here.
```

然后，在 syslog 的滚动日志配置文件/etc/logrotate.d/syslog 中，添加 iptables 的日志文件/var/log/iptables，详细内容如下：

```
/var/log/iptables /var/log/messages /var/log/secure /var/log/maillog
/var/log/spooler /var/log/boot.log /var/log/cron {
 sharedscripts
 postrotate
 /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
 endscript
}
```

最后安排 logrotate 每天执行一次，确定文件/etc/cron.daily/logrotate 内容如下：

```
#!/bin/sh
/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
/usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit 0
```

设置完成。

7.14.3 输出 iptables 日志到指定文件

本小节讲解对 iptables 进行配置，生成的 iptables 日志可以由 syslogd 记录和管理。iptables 日志初始存放在/var/log/messages 里面，但是由于混在 messages 中，可我们只需要其中的一部分，这对于管理产生了不便，一些情况下可能需要修改日志输出的位置。

iptables 有三种 log 记录形式，分别是 log、ulog、nflog。log 用于将匹配的数据包记录到系统的 syslog 中去，用户也可以直接通过 dmesg 命令查看。log 命令只记录包头的一些。ulog 通过 netlink 套接字将数据包多播到指定 netlink 多播组，这样任何感兴趣的进程都可以通过建立 netlink 套接字来接受内核中的数据包信息。ulog 可以将整个数据包复制并发送给应用程序，当然也可以指定发送方数据包的字节数。nflog 类似于 ulog 但功能更加强大，本节不讨论 nflog

的使用。

下面向大家介绍如何建立一个新的日志文件/var/log/iptables.log。通过修改或使用新的日志文件，可以创建更好的统计信息或者帮助我们分析网络攻击信息。

（1）iptables 默认的日志文件

例如，如果输入下面的命令，屏幕将显示/var/log/messages 文件中的 iptables 日志信息：

```
# tail -f /var/log/messages
```

输出为：

```
Oct 4 00:44:28 debian gconfd (vivek-4435): Resolved address
"xml:readonly:/etc/gconf/gconf. xml.defaults" to a read-only configuration source
at position 2
Oct 4 01:14:19 debian kernel: IN=ra0 OUT=
MAC=00:17:9a:0a:f6:44:00:08:5c:00:00:01: 08:00 SRC=200.142.84.36 DST=192.168.1.2
LEN=60 TOS=0x00 PREC=0x00 TTL=51 ID=18374 DF PROTO=TCP SPT=46040 DPT=22 WINDOW=5840
RES=0x00 SYN URGP=0
```

（2）将 iptables 日志转发到指定文件

在/var/log/手动创建 iptables.log 文件，然后打开/etc/syslog.conf 文件或者/etc/rsyslog.conf，这里以前者为例。

```
# vi /etc/syslog.conf
```

在文件末尾加入下面一行信息：

```
kern.warn  /var/log/iptables.log            \\注意不要尝试 kern.debug 日志类型，这
种报警数量很大。
```

然后保存退出，并重新启动 syslog 服务。

接着，设置 iptables 使用 log-level 4 参数（前面有一个 log-prefix 标志），例如：

```
#iptables -A INPUT -j LOG -log-level 4
#iptables -A INPUT -j DROP
```

接着查看 iptables 规则匹配：

```
#iptables L -n --line-numbers
```

举例，丢弃和记录所有来自 IP 地址 65.55.11.22 的连接信息到/var/log/iptables.log 文件。

```
#iptables -A INPUT -s 64.55.11.22 -m limit --limit 5/m --limit-burst 7 -j LOG
-log-prefix '** TEXT **' --log-level 4
#iptables -A INPUT -s 64.55.11.22 -j DROP
```

这个例子使用的命令解释如下：

- log-level 4：记录的级别，由表 7-2 可知级别 4 为警告（warning）。

● log-prefix '*** TEXT ***'：这里定义了在日志输出信息前加上 TEXT 前缀（也可输入其他内容）。TEXT 信息可超 16 个字符，这样就可以在记录文件中方便找到相关的信息。

现在可以通过/var/log/iptables.log 文件查看 iptables 的所有信息：

```
# tail -f /var/log/iptables.log
Jan 2 11:15:35 Chowroc kernel: FORWARD_o_ppp0: IN=eth0 OUT=ppp0 SRC=192.168.1.4
DST=23.15.62.82 LEN=64 TOS=0x10 PREC=0x00 TTL=63 ID=26229 DF PROTO=TCP SPT=1027
DPT=80 WINDOW=5840 RES=0x00 ACK URGP=0
```

如果需要将客户机上 iptables 转发到 OSSIM 服务器，只需要在 syslog.conf 中加入：

```
kern.warn      @Ossim Sensor IP
```

这里 facility 的[kern]和 priority 的[warn]，第一节已经讲过。为什么不用*.* @IP 呢?你必须清楚要收集特定的重要日志还是收集所有日志，很多用户漫无目的地收集日志，这样只会对后续的日志分析系统造成负担。

保存退出，然后重启 Syslog 服务，默认 OSSIM 启用防火墙，需要添加如下规则，日志才能够发送成功：

```
VirtualUSMAllInOne:/etc/iptables# cat rules012-custom.iptables
-A INPUT -p tcp -m state --state NEW -m tcp --dport 514 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```

为了将它的输出存放到指定的文件，还有一种方法就是使用 ULOG 扩展目标。iptables 的 ULOG 扩展目标用 netlink 多播组（multicast group）向用户空间发送捕捉到的消息。在 Debian 系统中安装 ULOG 很容易，输入以下指令：

```
#apt-get install ulogd
```

安装成功后，接着配置 ULOG，在 Debian 系统中，配置文件位于/etc/ulogd.conf 中，接着我们将多播组组号改成 32，位于该配置文件 11 行位置。

```
将"nlgroup=1"改成"nlgroup=32"
```

接着，开始一试身手吧，ULOG 目标在 iptables 中的典型用法如下：

```
#iptables -I OUTPUT -d 192.168.11.1 -j ULOG --ulog-nlgroup 1 --ulog-prefix
"TEXT:" --ulog-cprange 100 --ulog-qthreshold 10
```

● 参数"--ulog-nlgroup 32"，这是一个多播组组号，其含义代表使用编号是 32 的消息池广播消息。这个号一定要和 ulogd.conf 中指定的号相对应，才能收到消息。
● 参数"--ulog-prefix"，使用方法和 LOG 的 prefix 一样，只是长度可以达到 32 个字符。
● 参数"--ulog-cprange 100"，指定每个包要向"ULOG 在用户空间的代理"发送的字节数，表示把整个包的前 100 个字节复制到用户空间记录下来，其中包含了这个包

头，还有一些包的引导数据。默认值是 0，表示复制整个包，不管它有多大。

- 参数“--ulog-qthreshold 10”，表示先在内核里积聚 10 个包，再把它们发送到用户空间里，它们会被看作同一个 netlink 的信息，目的是告诉 ULOG 在向用户空间发送数据以供记录之前，要在内核里收集的包的数量。

这几个参数不是必须填入，用户可根据需要选择，由 ULOG 记录的日志位置在/var/log/ulog/syslogemu.log 文件中。

然后，在 Alienvault Center 中的插件管理中，加入 iptables 插件对日志进行归一化处理后，再提供给用户在 SIEM 控制台上浏览。

7.15 Squid 服务日志分析

Apache 和 Squid 是两种著名的代理缓存软件，但 Squid 较 Apache 而言是专门的代理缓存服务器软件，其代理缓存的功能强大，支持 HTTP/1.1 协议，其缓存对象也较多，并且 Squid 的缓存管理模块和访问控制模块功能很强大。所以在分析完 Apache 日志后，再看 Squid 日志就容易多了。最后，通过日志不但能分析 Squid 的基本的运行状态，还能在计算机取证中获取极具参考价值的信息。

7.15.1 Squid 日志分类

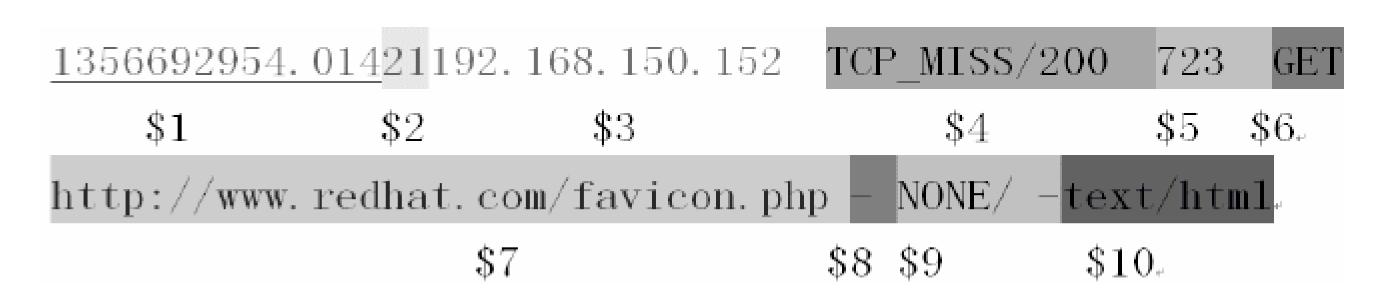
Squid 的日志系统相对比较完善，其主要日志分为两个：access.log 和 cache.log。它们的作用如下：

- access.log：客户端使用代理服务器的记录文件，访问日志位置在 squid.conf 中修改。
- cache.log：缓存在运行时的状态信息和调试信息，一般情况下容量不大。缓存日志位置在 squid.conf 中修改。

当代理服务器运行时，所有客户提出的请求，以及 Squid 处理的结果都会被记录在/var/log/squid/access.log 文件里，使得 access.log 文件的增长速度很快，通常会挂载一个较大的磁盘作为存储空间。

7.15.2 典型 Squid 访问日志分析

下面给出一条典型的 Squid 访问日志：



对这条日志的分析见表 7-7。

表 7-7　Squid 日志格式

域	值	含义
$1	1356692954.014	时间戳（记录了访问时间）
$2	21	持续时间
$3	192.168.150.152	IP 地址
$4	TCP_MISS/200	结果/状态码，正斜杠前表示 Squid 的结果码，后面表示状态码
$5	723	传输容量指传给客户端的字节数
$6	GET	请求方式
$7	http://www.redhat.com/favicon.php	URL
$8	-	客户端的 IDENT 查询一般为关闭
$9	NONE/ -	代码等级
$10	text/html	HTTP 请求头部

结果码 TCP_MISS 表示没有命中缓存，TCP_HIT 表示命中。

下面通过一个非常实用的 Shell 命令获取比较详细的命中情况：

```
# cat access.log|awk '{print $4}'|sort|uniq -c|sort -nr
33 TCP_MISS/200
  2 TCP_MISS/302
  2 TCP_MEM_HIT/302
  1 TCP_MISS/503
```

当然状态信息（TCP_MISS、TCP_MEM 等）不止这几个。总的来说，HIT 表示命中，而 TCP_MISS 表示未命中。

下列标签可能出现在 access.log 文件的第四个域。

- TCP_HIT：Squid 发现请求资源最新的复制，并立即发送到客户端。
- TCP_MISS：Squid 没有请求资源的 cache 复制。
- TCP_REFERSH_HIT：Squid 发现请求资源旧复制，并发送确认请求到原始服务器。
- TCP_IMS_HIT：客户端发送确认请求，Squid 发送更新的内容到客户端，而不联系原始服务器。
- TCP_NEGATIVE_HIT：在对原始服务器的请求导致 HTTP 错误时，Squid 会缓存这个响应。在短时间内对这些资源的重复请求，导致了是否命中。negative_ttl 指令控制这些错误被 cache 的时间数量。
- TCP_MEM_HIT：Squid 在内存 cache 里发现请求资源的有效复制，并将其立即发送到客户端。
- TCP_DENIED：因为 http_access 或 http_reply_access 规则，客户端的请求被拒绝了。
- TCP_REDIRECT：重定向程序告诉 Squid 产生一个 HTTP 重定向到新的 URI。正常情况下，Squid 不会记录这些重定向。

7.15.3　Squid 时间戳转换

（1）在表 7-7 所示的例子中，Squid 默认时间戳（1356693954.014）并不直观，下面通过脚本将时间戳换算成我们能识别的时间格式：

```
#perl -pe 's/^\d+\.\d+/localtime($&)/e;' access.log
```

如下所示，经过 perl 变化后的时间非常直观地显示出来，便于查看。

```
Fri Dec 28 22:05:30 2012    118 192.168.150.148 TCP_MISS/200 3705 GET http://safebrowsing-cache.google.com/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhchAAGMPiDyDM4g8yBkPxAwD_Aw - DIRECT/74.125.31.102 application/vnd.google.safebrowsing-chunk
Fri Dec 28 22:05:30 2012     74 192.168.150.148 TCP_MISS/200 1133 GET http://safebrowsing-cache.google.com/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhchAAGM3iDyDg4g8qB1DxAwD__wEyBU3xAwAH - DIRECT/74.125.31.102 application/vnd.google.safebrowsing-chunk
Fri Dec 28 22:05:49 2012    495 192.168.150.148 TCP_MEM_HIT/302 846 GET http://en-us.fxfeeds.mozilla.com/en US/firefox/headlines.xml - NONE/- text/html
Fri Dec 28 22:05:49 2012     32 192.168.150.148 TCP_MEM_HIT/302 890 GET http://fxfeeds.mozilla.com/firefox/headlines.xml - NONE/- text/html
Fri Dec 28 22:06:53 2012  63859 192.168.150.148 TCP_MISS/503 1566 GET http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml - DIRECT/59.24.3.173 text/html
```

（2）将 Squid 输出日志格式变形的脚本。

有时需要动态显示 squid 日志的第 3、8、7 列内容，以便更符合我们日常浏览习惯，就可以使用如下命令：

```
# tail -f /var/log/squid/access.log |awk '{print$3 "" $8""$7}'
192.168.150.148-http://safebrowsing-cache.google.com/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhchAAGMPiDyDM4g8yBkPxAwD_Aw
192.168.150.148-http://safebrowsing-cache.google.com/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhchAAGM3iDyDg4g8qB1DxAwD__wEyBU3xAwAH
192.168.150.148-http://en-us.fxfeeds.mozilla.com/en-US/firefox/headlines.xml
192.168.150.148-http://fxfeeds.mozilla.com/firefox/headlines.xml
192.168.150.148-http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xm
```

（3）可以将一个 squid 日志记录行分割成多个字段，使用参数传回需要的字段。

```
# tail -f /var/log/squid/access.log | awk '{print$3 " " $8 " " $7}'
```

这里选择的是客户 IP 及取回内容字段，显示如下：

```
192.168.150.146-http://jump.qq.com/clienturl_simp_80192.168.150.147 -
http://mm.china.com/zh_cn/images/tit_liangzhuang.gif192.168.150.148 -
http://ly.zzip.com.cn/movie/list.aspx?
```

（4）还可以根据日志分析命中率：

```
#cat access.log|awk '{print $4}'|sort|uniq -c|sort -nr
9568 TCP_IMS_HIT/304
6313 TCP_HIT/200
2133 TCP_MISS/200
1568 TCP_MISS/206
587 TCP_MEM_HIT/200
```

Squid 日志轮循方法可参考 Apache 中处理轮询方法。

7.15.4　将 Squid 的日志收集到 OSSIM

将 Squid 的日志收集到 OSSIM 步骤如下：

（1）首先将 Squid 插件加载到“detector”中，我们还要修改/etc/ossim/ossim_setup.conf 配置文件。

举例：

```
[sensor]
 detectors=snare, p0f, osiris, arpwatch, snortunified, pads,
 ssh, pam_unix, rrd, sudo, iptables, nagios, squid
```

（2）修改 squid 插件内容。

编辑/etc/ossim/agent/plugins/squid.cfg 插件配置文件，将原有配置按照下面例子修改，squid 的插件 ID 号为 1553 。

```
create_file=true  \\*这行配置代表如果不存在这个文件就创建*\\
process=squid
 start=yes ; launch plugin process when agent starts
 stop=yes ; shutdown plugin process when agent stops
 startup=/etc/init.d/%(process)s start
 shutdown=/etc/init.d/%(process)s stop
 restart=yes ; restart plugin process after each interval
 restart_interval=\_CFG(watchdog,restart_interval) ; interval between each
restart
```

（3）重配置 OSSIM。

```
#/usr/bin/ossim-reconfig
```

这时会产生/var/log/squid/access.log 日志文件：

```
#tail -f /var/log/squid/access.log
Aug 12 17:26:00 ossim squid[11680]: 1282814160.311   1291 192.168.150.219
TCP_MISS/200 405 POST
http://164.24.134.107/gateway/gateway.dll?Action=poll&SessionID=989804211.9675
03471 - DIRECT/164.24.134.107 application/x-msn-messenger
Aug 12 17:26:20 ossim squid[11680]: 1282814180.328   1299 192.168.150.219
TCP_MISS/200 404 POST
http://164.24.134.107/gateway/gateway.dll?Action=poll&SessionID=989804211.1408
535067 - DIRECT/164.24.134.107 application/x-msn-messenger
```

7.16 DHCP 服务器日志

DHCP（Dynamic Host Configuration Protocol，动态主机配置协议）是一种有效的 IP 地址分配手段，现已经被广泛地应用在各种局域网管理中。它能动态地向网络中的每台计算机分配唯一的 IP 地址，并提供安全、可靠、简单和统一的 TCP/IP 网络配置，确保不发生 IP 地址冲突。当在服务器上启用 DHCP 后，我们希望了解服务的运行情况，希望看到详细日志。可以通过下面的命令了解到 dhcp server 的日志文件在什么地方。以 RHEL5 系统为例，命令如下：

```
#rpm -ql dhcp-server
```

DHCP 服务的默认日志不会输出到指定文件，而是和 NFS 服务一样，输出到/var/log/messages 文件中，成了日志的“大杂烩”，既不方便识别，也不便于查找故障，一旦 messages 文件遭到破坏，DHCP 的日志也跟着受影响。

```
Dec 31 16:32:51 localhost dhcpcd[5562]: br0: trying to use old lease in `/var/lib/dhcpcd/dhcpcd-br0.info'
Dec 31 16:32:51 localhost dhcpcd[5562]: br0: lease expired 4041 seconds ago
Dec 31 16:32:51 localhost dhcpcd[5562]: br0: broadcasting for a lease
Dec 31 16:32:51 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:32:51 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:32:54 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:32:54 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:32:57 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:32:57 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:00 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:00 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:03 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:03 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:06 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:06 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:09 localhost dhcpd: DHCPDISCOVER from 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:09 localhost dhcpd: DHCPOFFER on 192.168.150.201 to 00:0c:29:51:b3:d9 (linux-5jlv) via br0
Dec 31 16:33:11 localhost dhcpcd[5562]: br0: timed out
Dec 31 16:33:11 localhost dhcpcd[5562]: br0: trying to use old lease in `/var/lib/dhcpcd/dhcpcd-br0.info'
Dec 31 16:33:11 localhost dhcpcd[5562]: br0: lease expired 4061 seconds ago
--More--
```

对于以上日志，我们可以把在前几节学到的脚本放到这里进行分析。还有没有其他文件，记录了 DHCP 的分配 IP 的信息呢？还有一个是/var/lib/dhcp/db/dhcpd.leases 文件，它记录了客户机分配 IP 的详细信息。下面我们通过一个例子进行解读。

客户机每次获取地址后会产生如下信息：

```
Lease 192.168.150.207 {
  Starts 1 2012/12/31 11:23:32
  End  1 2012/12/31 11:25:32;
  Tstp  1 2012/12/31 11:25:32;
  Cltt  1 2012/12/31 11:25:32;
  Binding state free;
  Hardware ethernet  00:0c:29:51:b3:d9;
  Uid "\001\000\014)Q\263\331";
 Client-hostname "linux-5jlv";
}
```

每当发生租约变化的时候，都会在文件结尾添加新的租约记录，也就是说这个文件是在不断变化的。表 7-8 做出解释。

表 7-8　DHCP 日志含义

参数	含义
Lease	租用 IP
starts	开始时间
end	结束时间
tstp	指定租约过期时间

（续表）

参数	含义
cltt	客户端续约时间
Binding state	租约绑定状态自由（free）、激活（active）
Hardware ethernet	客户机网卡 MAC 地址
UID	客户端标识符由三位八进制表示用于与 MAC 匹配
Client-hostname	客户机名称

从上面分析看到，DHCP 服务器的日志在 messages 和 dhcpd.leases 里分别有一部分，都不全面，如何将 DHCP 的日志专门转储到特定文件中呢？下面介绍一种方法。

假设需要将日志记录在/var/log/目录下，我们先新建一个 dhcp.log 文件。

（1）创建 dhcp.log 文件：

```
#touch /var/log/dhcp.log
#chmod 640 /var/log/dhcp.log
```

（2）修改/etc/dhcpd.conf 配置文件，然后保存并退出（注意不同 Linux 发行版配置文件路径有所不同）：

```
log-facility local4;
```

（3）在/etc/rsyslog.conf 文件中添加：

```
Local4.* /var/log/dhcp.log
```

注意要把下面这行语句注销：

```
Local4,local5.*  -var/log/localmessages;RSYSLOG_TraditionalFileFormat
```

重启 DHCP 服务即可生效，这时的日志文件就是 DHCP 服务器出现故障后，排除错误的一个重要基础数据。所以，我们还需要定期对这个日志文件做好备份工作。否则，如果这个日志意外丢失，我们就很难查清 DHCP 服务器的故障。

7.17 收集 Windows 日志

OSSIM 是目前为数不多的几个开源的 SIEM/安全管理平台之一，而目前还没有什么集成化的日志管理分析（称日志管理系统，简称 LM）系统，OSSIM 是比较好用的一种 LM，还有 Sawmill 和 Splunk 都有针对 UNIX/Linux 的版本。Agent 这个概念在 OSSIM 日志收集系统中非常重要，因为在系统的日志收集过程中，它非常适合大型网络中的分布式日志收集工作。分布式日志收集架构如图 7-40 所示。

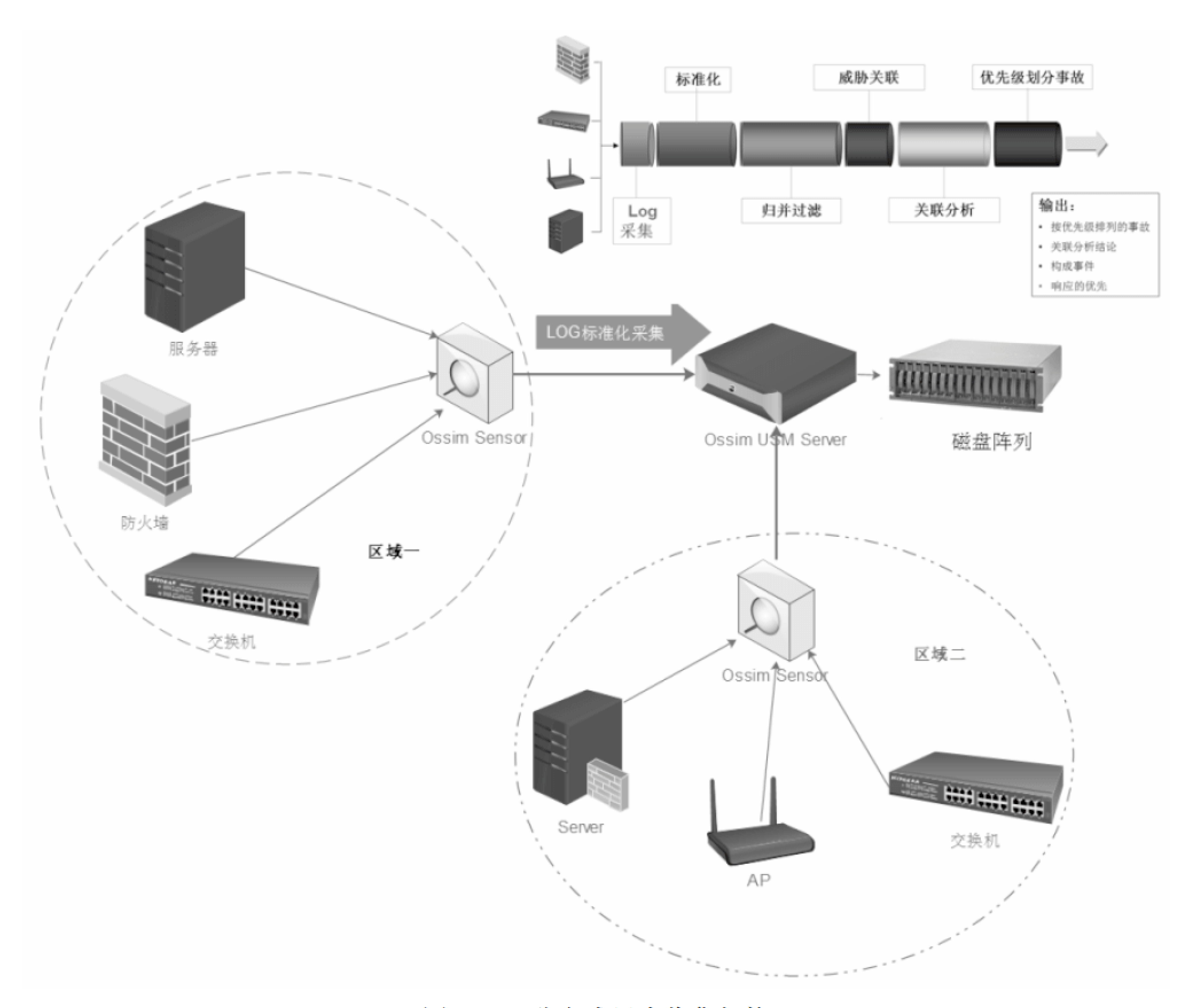


图 7-40　分布式日志收集架构

从上图中我们知道，安装在终端上的常用代理既包括 syslog，还包括 Snare（系统入侵分析和报告环境），Snare 是一款开源软件。Snare 逻辑上由内核动态加载模块、用户空间审核程序以及 Web 分析前端三部分组成。通过图形化的日志分析前端便可使用户得到可自定义的、规范的系统日志和事件报告。

7.17.1　OSSIM 日志处理流程

首先设备把日志信息以 Syslog 的形式发给 Agent，日志存储在/var/log/下，如果是 Snort，则日志位置为/var/log/snort.log。Agent 程序则会调用/etc/ossim/agent/plugins 下面对应的 Snort 插件来/var/log/snort.log 下面取对应的日志，然后根据插件里面写的正则表达式来提取日志的关键字段发给 Server 端，最后由 Server 再将日志分析之后在 OSSIM 上面呈现出来。

7.17.2　通过 Snare 转发 Windows 日志

Windows 日志可记录应用程序、安全和系统事件，具备一定的安全性，它采用二进制方式记录，格式较复杂，不易直接查看，Snare for Windows 可以把 Windows 系统事件日志实时

转发到 SYSLOG 服务器上，它可用于发现和检测 Windows 系统异常，它支持的日志类型有安全日志、应用日志、系统日志，以及活动目录（Active Directory）日志等。下载安装 Snare for Windows 的位置（需要下载代理，很显然它是基于 Windows 的代理程序）在 OSSIM 4.3 系统左侧菜单中，依次单击 Deployment→Collection→Downloads，在 OSSIM 4.8 之后的版本，下载位置在 Web UI 的右上角“SUPPORT”按钮处。在 Windows 下安装此程序非常简单，安装过程中只要使用系统账户安装即可。安装完毕，可以在开始菜单中看到以下三个菜单栏：

- Disable Remote Access to Snare for Windows：关闭 Snare 的远程管理；
- Restore Remote Access to Snare for Windows：恢复 Snare 的远程管理；
- Snare for Windows：程序配置界面，如图 7-41 所示。

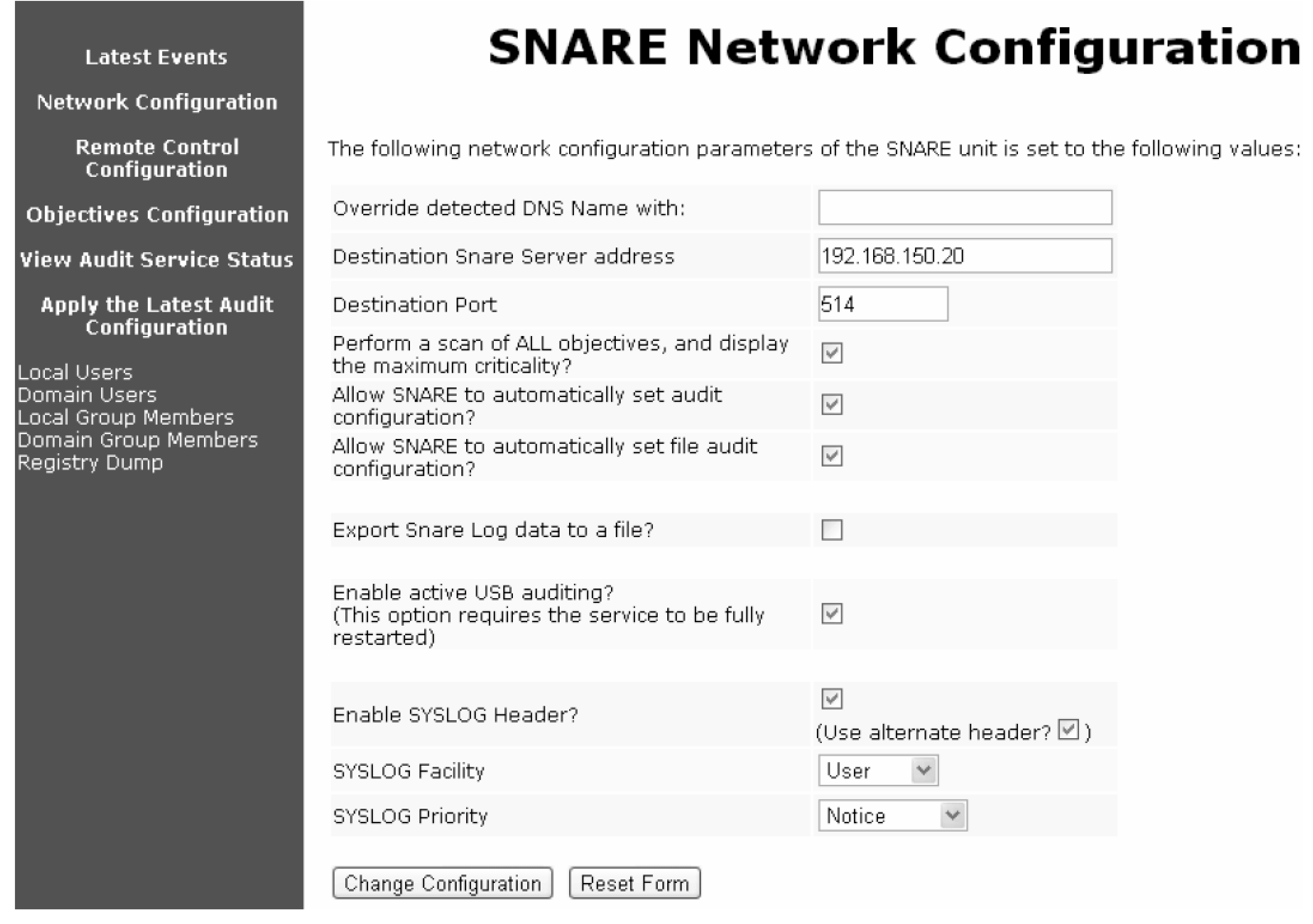


图 7-41　Snare 管理配置

初次装完 Snare for windows 代理之后，首先在 Windows 系统中，选择程序→InterSect Alliance，再选择 Restore Remote Access to Snare for Windows，然后即可通过 Snare for Windows，打开一个 Web UI 界面，进入后发现，Destination Snare Server address 为 127.0.0.1，将它改为 OSSIM 服务器地址 192.168.150.20，而 Destination Port 初始值为 6161，将它改为 514，同时选择以下复选框：

- Perform a scan of ALL objectives, and display the maximum criticality?(执行所有目标的

扫描，并显示最关键日志）

- Allow SNARE to automatically set audit configuration?（允许 Snare 设置为自动审核配置功能）
- Allow SNARE to automatically set file audit configuration?（允许 Snare 设置为自动文件审核配置功能）
- Enable active USB auditing?(This option requires the service to be fully restarted)（启用 USB 审核功能？该选项需要重启服务）
- Enable SYSLOG Header?（启用 Syslog Header 功能，完整 Syslog 消息包括 PRI、HEADER 和 MSG，有些情况下 HEADER 可能没有，这里指强制加入 HEADER）

SYSLOG Facility 和 Priority 也为必选，他们包含关系见表 7-9 所示，另外，Overrid detected DNS Name with 和 enable active USB auditing 为可选项。

表 7-9　Facility 和 Priority 内容对比

SYSLOG　Facility	Kernel User Mail Daemon Auth Syslog Lpr news Uucp Cron Authpriv ftp Local 0 ~ local7	SYSLOG Priority	Emergency Alert Critical Error Warning Notice Information Debug Dynamic

首先，要确保 Snare 管理为打开状态（在 Windows Vista 以上系统，使用要注意以管理员身份运行，否则会出现启动错误），在浏览器打开 http://localhost:6161/地址，选择左侧菜单的 Network Configuration 选项。

然后在“Destination Snare Server address”地址栏填写 OSSIM 的地址，目标端口为 514。这时就可以在 OSSIM 控制台上接收到 Windows 服务器发来的日志了。配置完毕，在左侧选择“Apply the latest Audit configuration”菜单，并单击“Reload Settings”按钮，重新加载设置。

最后，在 OSSIM Sensor 上启用 Snare 插件（参考插件配置图 7-19）。

操作举例，配置 Windows Snare 日志步骤如下：

（1）在 Windows 客户机上安装并配置 Snare，这里假设 OSSIM 服务器 IP 为 192.168.150.20，主机名为 alienvault。

（2）在主机 alienvault 上修改/etc/hosts，添加 Windows 主机名和 IP 的映射。

（3）当 Windows 上的 Snare 装好后，在 Alienvault 中重启 Agent 进程。

```
#/etc/init.d/ossim-agent restart
```

（4）打开/etc/ossim/agent/plugins/snare.conf 配置进行验证。

确保 snare.conf 存储的日志在/var/log/snare.log 文件中（默认的 location 为/var/log/syslog），如果没有自动创建 snare.log，将创建文件选项由 false 改成 true，即 create_file=true。

（5）新建配置文件/etc/rsyslog.d/snare.conf，加入以下几行内容。

```
if $msg contains 'alienvault' then -/var/log/snare.log
if $msg contains '192.168.150.20' then -/var/log/snare.log
if $msg contains 'MSWinEventLog' then -/var/log/snare.log
if $fromhost-ip == '192.168.150.20' then /var/log/snare.log
if $rawmsg contains 'MSWinEventLog' then /var/log/snare.log
& ~
```

重启 Rsyslog 服务。

```
#/etc/init.d/rsyslog reload
```

在 Windows 客户端导入注册表文件，稍等片刻后再查看 snare.log，即能收集到日志，实例如下：

```
#tail -f /var/log/snare.log
Nov 15 11:21:31 alienvault.redacted MSWinEventLog;0;Security;178;Thu Nov 15
11:21:29
2012;4689;Microsoft-Windows-Security-Auditing;TST\alienvault$;N/A;Success
Audit;alienvault.redacted;Process Termination;;A process has exited.   Subject:
Security ID:  S-1-5-18   Account Name:  alienvault$   Account Domain:  TST   Logon
ID:  0x3e7    Process Information:   Process ID: 0xb3c   Process Name:
C:\Windows\System32\wbem\WmiPrvSE.exe   Exit Status: 0x0;74
Nov 15 11:22:42 alienvault.redacted MSWinEventLog;1;System;179;Thu Nov 15
11:22:41 2012;7036;Service Control
Manager;N/A;N/A;Information;alienvault.redacted;None;;The Application
Information service entered the running state.;80
```

为了调试需要，建议大家在 OSSIM 服务器端的命令行下监视所有发往主机 192.168.150.20 514 端口的数据包，操作如下：

```
#tcpdump -i eth0 host 192.168.150.20 and port 514 - v
```

除了 Snare 工具以外，evtsys 这款工具也可将 Windows 日志发送至 Syslog 服务器。还包括 NTsyslog，它们都能以系统服务的方式运行，都具有软件小巧、运行高效等特点。有关 Snare 的更多内容大家可以到 http://www.intersectalliance.com/网站中继续学习。在 SIEM 控制台中查看收集到 Windows 日志如图 7-42 所示，在 RawLog 中查看收集到的 windows 日志，如图 7-43 所示。

图 7-42　收集到 Windows 日志

图 7-43　在 RawLog 中查看收集到的 Windows 日志

7.17.3　通过 WMI 收集 Windows 日志

Microsoft Windows 管理规范，简称 WMI。通过企业网络为访问和共享管理信息主动建立标准. WMI 提供程序在 WMI 和操作系统和应用程序之间充当中介角色, 所以可以通过 WMI 来检索大多数计算机系统方面配置的详细信息。WMI 提供程序主要包括以下内容：

- 为开发人员提供硬件类、系统类和进程管理类的类库。
- 安全提供程序，主要用户安全设置（所有权、审计和访问权限）。
- 事件日志提供程序，提供对 Windows 事件日志的访问，例如读取、备份更改事件日志设置等。
- 性能计数器和监控器提供程序，负责读取、写入及监视。
- SNMP 负责提供对 SNMP MIB 数据的访问，并从 SNMP 托管设备获取信息。

首先确保在 Windows 下 WMI 服务是启动状态，然后在 Windows 开始菜单的命令行中输

入“DCOMCNFG”命令调出组件服务，然后依次单击控制台根目录→组件服务→计算机→我的电脑，最后在“我的电脑”上右键单击属性，单击“COM 安全”标签，然后再启用和激活权限栏目，单击编辑限制，如图 7-44 所示。

图 7-44　设置 WMI

成功添加 wmiuser（事先添加该用户）用户后，即完成了 Windows 系统上的设置工作。

7.17.4　配置 OSSIM

在 OSSIM 系统中安装了 WMI 插件后，需要在 Windows 做好相应设置，刚才我们已经完成，下面还要对 OSSIM 的配置文件做一些调整。

首先在/etc/ossim/agent 目录下创建 wmi_credentials.csv 文件：

```
#vi /etc/ossim/agent/wmi_credentials.csv
```

然后添加若干台 Windows 计算机，包括 IP 地址用户名称和密码，格式如下：

```
192.168.150.10,user1,pass
192.168.150.11,user2,pass
```

本实验在 Windows 域环境下出现过某些 Windows 机器无法发出日志的情况，建议大家使用工作组内的 Windows 机器，而且别忘了在调试期间，关闭 Windows 防火墙。

最后开始激活 WMI 插件，方法如下：

（1）执行 ossim-setup。

（2）选择第 3 项 Change Sensor Settings。

（3）选择第 3 项 Select detector plugins。

（4）选择 wmi-application-logger。

（5）选择 wmi-system-logger。

（6）选择 wmi-security-logger。

（7）保存并退出，紧接着系统启动 ossim-reconfig 开始重新配置系统。

（8）重新启动代理进程/etc/init.d/ossim-agent restart。

很快就可以在 SIEM 中收到标记 Snare Windows 的日志了。

当收不到日志时请稍安勿躁，先尝试以下命令测试连接。

```
#tail -f /var/log/ossim/agent.log
2012-11-04 10:08:45,088 Detector [INFO]: Starting detector wmi-system-logger (1518)..
2012-11-04 10:08:45,186 Detector [INFO]: Starting detector wmi-application-logger (1518)..
2012-11-04 10:08:45,711 Detector [INFO]: Starting detector snare (1518)..
2012-11-04 10:08:45,795 Detector [INFO]: Starting detector wmi-security-logger (1518)..
2012-11-04 10:08:46,694 ParserWMI [INFO]: [1518] Section found, last record : 0
2012-11-04 10:08:46,701 ParserWMI [INFO]: [1518] Section found, last record : 0
```

这条命令的含义是检查代理是否收到日志。接着在 Windows 系统下输入：

```
C:\>wmic -U<user>%<pass>//<192.168.150.20>"select * from win32_Process"
```

其中 wmic 是 Windows 管理规范的命令行工具，最早随 Windows Server 2003 发布，这条命令的含义是检查与 Windows 机器是否连接。

```
#tail -f /var/log/alienvault/server/server.log
```

这条命令用于检验服务器是否收到日志，我们可以在 Web 界面查看配置情况，如图 7-45 所示。

图 7-45　Web 下配置 WMI

7.17.5　Snare 与 WMI 的区别

我们知道 Windows 系统的图形界面非常强大而且易用，但是图形界面需要较大的资源消

耗，使得一些系统维护人员不太满意，所以微软开发了 WMI，在其中的 Resource Kits 提供了大量基于 WMI 的脚本供管理员使用，WMI 通过 RPC 调用访问 Windows 系统的原始数据，所以 WMI 对 Windows 系统支持得最好，获取的日志信息也最完整。

前面介绍过在 UNIX/Linux 和一些路由器交换设备上会产生大量日志信息，并以 syslog 的形式存在，打个比方这个 syslog 日志协议，可以告诉管理员谁（Facility），什么时间（Timestamp），什么地方（Hostname）做了什么事情（Message），以及这个事情的重要性（Severity）。在 Windows 系统中没有使用 syslog 协议去收集日志，因为它有自己的日志协议 Event Log。

Snare 是一个代理程序，Snare 可以将 Windows 事件日志转发到 syslog 服务器中，并且它没有 32 位和 64 位之分，它将 Windows 日志转发到 OSSIM 系统的 syslog 服务上并由它接收。不仅是 OSSIM 系统利用 WMI 收集 Windows 日志，Splunk、ManageEngine Eventlog Analyzer、Sawmill 等日志分析系统亦是如此。

7.18 小结

本章详细介绍了常见日志格式以及收集标准，并分析了路由器、交换机、防火墙常见网络设备的日志，除此之外还详细讲解了 Linux 平台下 Apache、Vsftp、Squid、DHCP 等应用服务日志的日志分析，最后讲解了 OSSIM 中 Snare 和 WMI 收集日志的流程和方法。

第 8 章 OSSIM流量分析与监控

从本章节可以学习到：

- 用 NetFlow 分析异常流量
- NetFlow 输出格式与保存方法
- Ntop 流量采集方式
- Ntop 流量分析方法
- Nagios 原理
- Nagios 监控方法
- Nagios 插件
- 第三方监控软件集成
- 硬件监控

8.1 用 NetFlow 分析异常流量

目前主机的异常流量主要由以下几类行为所造成：

（1）网络蠕虫、病毒

现今网络病毒和蠕虫的传播，导致网络带宽或主机和网络设备资源的极大浪费。

（2）DOS 和 DDoS 攻击

拒绝服务攻击流量巨大，常常会破坏主机或网络的可用性，DOS 攻击常使用异常的数据流量冲击主机或网络设备，尤其是分布式拒绝服务攻击可以控制多台主机同时发起攻击，造成攻击目标崩溃。

（3）其他入侵引起的异常流量

除了蠕虫和 DOS 攻击外，还有 Shellcode 攻击、缓冲区溢出（ARP）攻击等引起的异常流量，这类异常更不好查找。不同的入侵行为会具有不同的异常流量特征，如突发性、小概率性等，使得主机的实时流量呈现多维度的特点。

这些流量可以使用 NetFlow 分析异常流量。NetFlow 流量分析法可帮助用户了解流量构成、协议分布，与传统基于 SNMP 的监控工具 Cacti、Zabbix 所不同，它利用 Flow 技术来收集网

络中有关流量的重要信息，在 OSSIM 系统中集成了 NetFlow 后，可以实现集流量收集、分析、报告于一体。该功能和 OSSIM 中的实时抓包分析形成了有利的相互补充。就好比一个病人去医院进行验血、拍 X 光片一样由表及里深入到内部查找问题。下面首先了解一下 NetFlow 基础知识，为更深入地学习打个基础。

最初 NetFlow 由 Cisco 开发，由于使用广泛，目前很多厂家都可以实现类似 NetFlow 的功能，如：Juniper、Extreme、Foundry、H3C。对于 Cisco 来说，NetFlow 有多种版本，如：V5、V7、V8、V9。目前 NetFlow V5 是主流。因此本文主要针对 NetFlow V5。首先从流（Flow）讲起，一个 IP 数据包的 Flow 至少定义了下面 7 个关键元素：

- 源 IP 地址;
- 目的 IP 地址;
- 源端口号;
- 目的端口号;
- 第三层协议的类型;
- TOS 字段;
- 网络设备输入/输出的逻辑端口。

以上 7 个字段定义了一个基本的 Flow 信息，不过 Cisco 的 Neflow v5 版本中还包含了 AS 字段. 在 OSSIM 系统中集成了一款基于 Web 的 NetFlow 分析工具,其中也是通过收集 NetFlow 中的以上这些信息来分析流量从而判断故障。

当前还可利用 NetFlow 或 sFlow 当中的一种，所不同的是，NetFlow 是一种基于软件的技术，而 sFlow 则采用内置在硬件中的专用芯片。这种技术减轻了路由器或交换机的 CPU 和内存的负担。无论是 NetFlow 还是 sFlow，都可以在无须部署探测器的情况下，帮助网络管理员更深入了解网络传输流。

8.1.1 流量采集对业务的影响

采用 NetFlow 方案处理从某个接口接收到数据包,用来对被监控路由器进行流量分析的数据来自 NetFlow 数据从路由器送出的非采样 NetFlow 数据不到流经该路由器数据量的 1%，使用采样 NetFlow 时数据大为减少。根据计算，在采样率为 1000:1 时，对 10Gbit/s 的流量进行 NetFlow 分析，约产生 1.3Mbit/s 的流量。因此，NetFlow 产生的这部分流量对于骨干网的带宽占用很少。

利用 NetFlow 技术实现流量监测需要路由器打开 NetFlow 协议，以配合采集数据。因此会对路由器的 CPU 造成一定的负担。根据 Cisco 公司的“NetFlow Performance Analysis”白皮书，在非采样方式下，路由器打开 NetFlow 后，其 CPU 使用状况如下：

- 如果有 10000 条同时在线的 Flow，则路由器的 CPU 使用率平均增加 7.04%;
- 如果有 45000 条同时在线的 Flow，则路由器的 CPU 使用率平均增加 19.06%;
- 如果有 65000 条同时在线的 Flow，则路由器的 CPU 使用率平均增加 21.08%。

这些数据是基于不同的产品系列进行测试的平均值。

在采样方式下打开 NetFlow 对路由器的 CPU 影响会更小。根据 Cisco 公司的资料显示，在采用 100:1 的采样率时 CPU 使用率仅增加 3%。在不同的采样率下，CPU 的负担增加程度也不同。

8.1.2　NetFlow 的 Cache 管理

在 NetFlow 中有两个关键组件：Cache 和 Expert。

（1）NetFlow Cache 主要描述流缓存是如何存放在 Cache 中的。

NetFlow 缓存管理机制中包含一系列算法，能够有效地判断一个报文是属于已存在 Flow 的一部分，还是应该在缓存中产生一条新的 Flow。这些算法能动态更新缓存中 Flow 的信息，并且判断哪些 Flow 应该到期终止。

（2）NetFlow Expert 主要描述流的输出机制，也就是如何输出并被分析器接收。

首先了解 NetFlow Cache（缓存机制）。当缓存中的 Flow 到期后，就产生一个将 Flow 输出的动作。将超时的 Flow 信息以数据报文的方式输出，这叫做“NetFlow Export”，这些输出的报文包含几十条 Flow 信息。

8.1.3　NetFlow 的输出格式

NetFlow 的输出报文包含报头和一系列 Flow 流，报头包含序列号、记录数、系统时间等，Flow 流包含 IP 地址、端口、路由信息等。各个版本的 NetFlow 格式都相同，且 NetFlow 采用 UDP 报文，更有利于大流量情况下的报文传输。

8.1.4　NetFlow 的采样机制

在 NetFlow 的实际应用中，它使用采样机制，通过使用采样技术可以降低路由器的 CPU 利用率，减少 Flow 的输出量，但仍然可以监测到大多数流量的基本状态信息。当我们不需要了解网络流量的每个 Flow 的细节时，采样就成了比较好的选择。但它不适用于计费系统，因为当流量计费系统采用 NetFlow 技术会造成误差。

8.1.5　NetFlow 采样过滤

多数异常流量的目的端口固定在一个或几个端口，我们可以利用这一点，对异常流量进行过滤或限制。传统方式下我们在路由器上启用 NetFlow，并登录该设备输入如下命令查看，如果是长期的数据分析，这种命令行方式不便于观察和分析故障。如图 8-1 所示。

```
GATEWAY#show ip cache flow
IP packet size distribution (1149 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .000 .134 .475 .100 .010 .006 .037 .043 .005 .001 .004 .001 .002 .001 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .003 .000 .001 .020 .147 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  13 active, 4083 inactive, 378 added
  7046 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 21640 bytes
  13 active, 1011 inactive, 378 added, 378 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-WWW             32      0.0         8   989      0.1       3.8       8.1
TCP-other           24      0.0         2    57      0.0       2.2      14.4
UDP-other          309      0.1         2   105      0.3       2.4      15.4
Total:             365      0.1         3   318      0.4       2.5      14.7

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP  Pkts
Fa0/0         10.0.0.23       Null          10.255.255.255  11 0089 0089     9
Fa0/0         10.0.0.30       Null          10.255.255.255  11 008A 008A     1
```

图 8-1　路由器上查询 NetFlow

而在 OSSIM 系统中提供的这种图形化的 NetFlow 模式过滤，不消耗路由器系统资源，如图 8-2、图 8-3 所示。

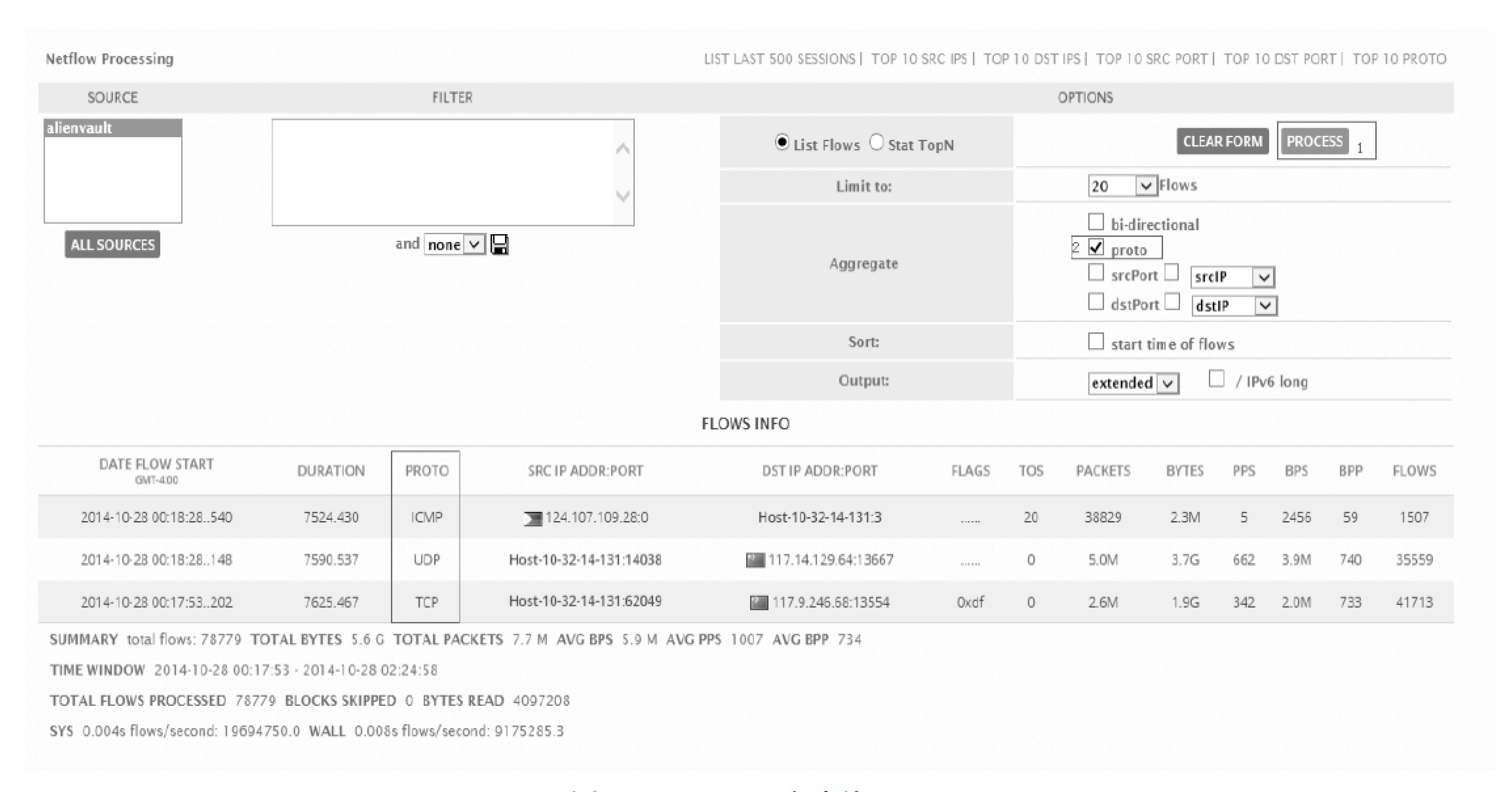


图 8-2　OSSIM 上查询 NetFlow

图 8-3　NetFlow 数据流过滤

8.2 NetFlow 在监测恶意代码中的优势

相对于传统的基于 Payload 的恶意代码检测方法而言，基于 NetFlow 的检测方法具有如下 3 点优势：

（1）NetFlow 数据流获取方便。

（2）NetFlow 流信息没有高层信息。在 OSSIM 的 SIEM 面板中出现的基于 Payload 的检测方法，由于需要分析应用层信息，所以不可避免地会降低分析程序的处理效率。而 NetFlow 并不包含应用层信息，但包含三层信息（含三层协议类型）。虽然在分析的精度上有所下降，但是使得分析程序的效率大为提高，尤其是在大流量环境下优势更为明显。

（3）现在 NetFlow 已经成为事实上的标准，而且 OSSIM 提供的友好的图形化 NetFlow 界面，简化了数据分析难度，主要是系统通过 RRD（Round-Robin Database）实现可视化，它将数据存储于 RRD 数据库，然后生成图片根据进出数据按端口、协议类型表现出来。如图 8-4 所示。我们可以显示全年流量，直观地查看流量分布情况。

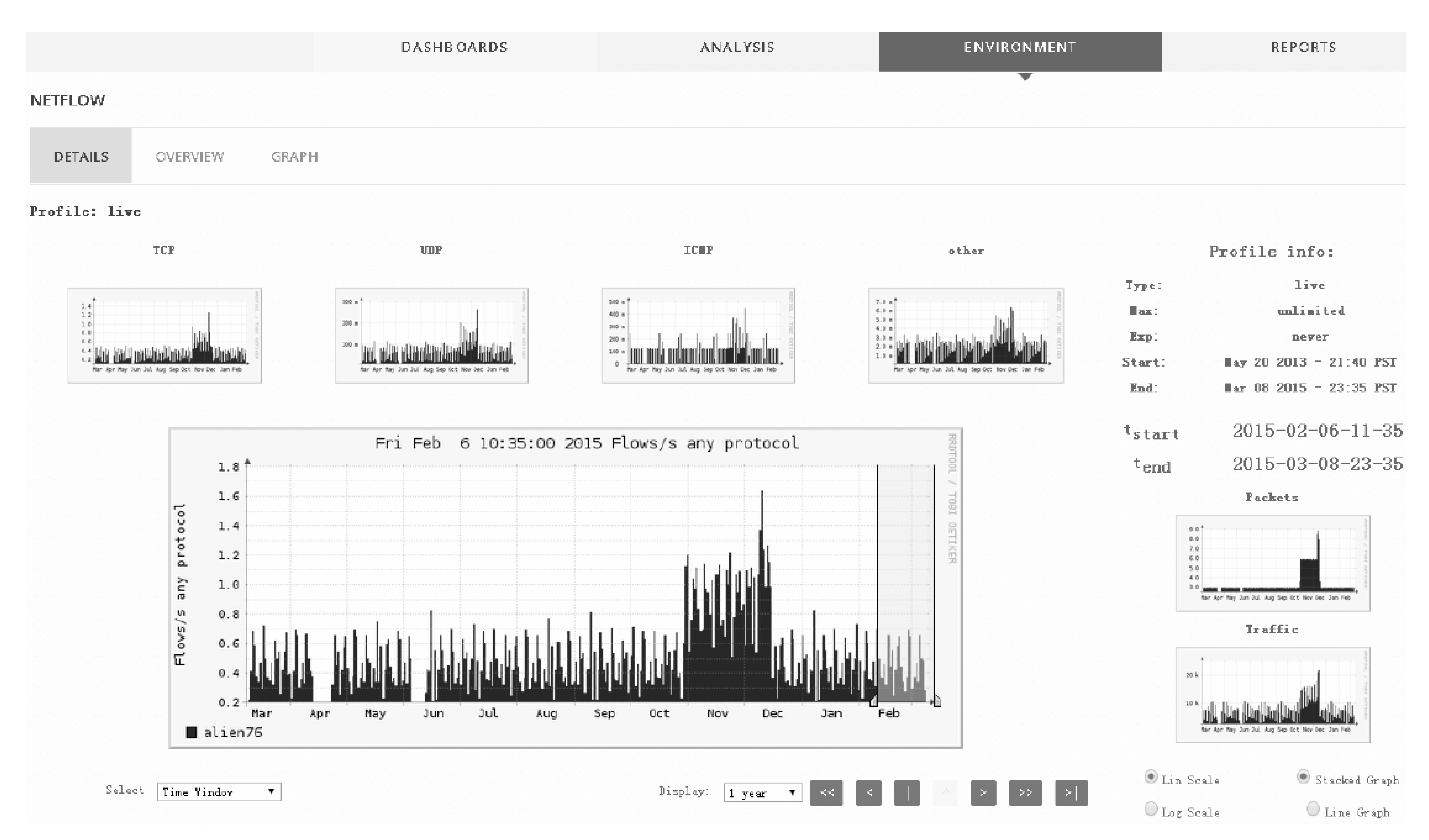


图 8-4　显示全年的 NetFlow 流量

8.2.1　NetFlow 的性能影响

我们知道在设备缓存中 Flow 的生成，需要消耗系统资源，将 Flow 格式化成特定的报文，并将报文输出，同样会消耗系统资源，因此在设备上使用 NetFlow 时，会牺牲设备性能。由于高端 Cisco 设备(如 6500、7600 系列等)都是通过 ASIC 硬件处理数据包，所以占用 CPU 10%～15%利用率都是正常。在使用中 CPU 的利用率随着缓存中 Flow 条目的增大而增加，所以在高负载情况下，需慎用 NetFlow 功能。

8.2.2　NetFlow 在蠕虫病毒监测的应用

前些年 Red Code、SQL Slammer、冲击波、震荡波等病毒的相继爆发，不但对用户主机造成影响，而且对网络正常运行也构成危害，因为这些病毒具有扫描网络、主动传播病毒的能力，还会大量占用网络带宽或网络设备系统资源。这些蠕虫在网络行为上都有某些共同特征，我们可以利用 NetFlow 筛选出带有这些特征的数据包，从而发现问题。其实最简单的检测扫描的算法就是看某个 IP 是否连续访问某个 IP 段内的所有主机。NetFlow 分析恶意软件的流程如图 8-5 所示。

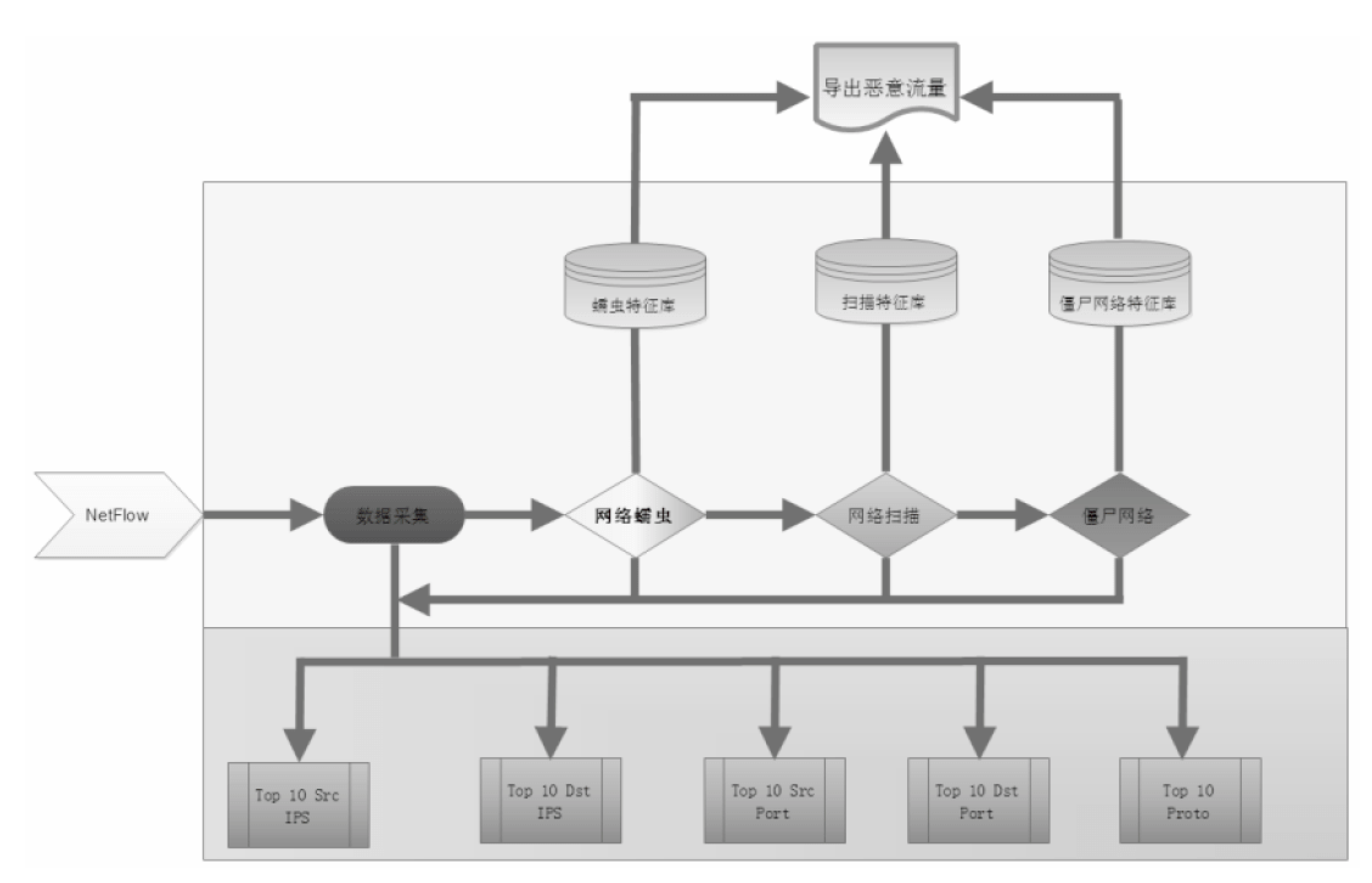


图 8-5 NetFlow 分析恶意软件的流程

例 1：CodeRed 的 Flow 特征是 destination port=80， packets=3， size=144bytes。虽然在 Internet 上，符合上述特性的正常行为是存在的，但是一般正常使用的主机不会连续发出大量的报文。

因此监测 CodeRed 可采用策略是：取几个不同时间段，例如每段时间 5 分钟，如果每个时间段内符合特征的 Flow 大于上限值，则可以判断为 CodeRed。

例 2：Nimda 的 Flow 特征是每个 Flow 代表一次连接 destination port=80（http）的行为，如果普通的客户机在一段时间内（例如 5 分钟内）Flow 数量过大，那么很有可能遭受病毒感染或者存在其他针对 HTTP 的攻击行为。

因此监测 Nimda 可采用策略是：选取几个不同时间段，间隔为 5 分钟，如果时间段内符合特征的 Flow 超过上限值，则可以判断为 Nimda 病毒攻击行为。

8.2.3 网络扫描和蠕虫检测的问题

蠕虫爆发的初始阶段会对网络主机进行扫描，进而感染更多的存在特定漏洞的主机。所以爆发初期蠕虫在流特征上和网络扫描类似。可在策略中定义一个经验值，当每个源 IP 对应的符合流特征的目标 IP 多于这个经验值时，就认为这是扫描流量，将源 IP、累计扫描次数、扫描种类放入黑名单表，以便在必要时，对其源 IP 进行过滤。

例 3：震荡波病毒（Worm.Sasser）的特征是一个 IP 同时向随机生成的多个 IP，发起 445 端口的 TCP 连接。因此检测条件是：相同源 IP、大量不同目的 IP、目的端口为 445，当符合

的 Flow 达到上限值时，则可以确定是震荡波病毒，它的 NetFlow 流如表 8-1 所示。

表 8-1　震荡波（W32.Sasser.Worm）NetFlow 流记录

时间	源 IP	源端口	目的 IP	目的端口	包	字节	标志	协议
2014-1-1 14:10	192.168.1.100	1395	16.203.2.1	445	1	48	0	TCP
2014-1-1 14:10	192.168.1.100	1288	78.21.24.15	445	1	48	0	TCP
2014-1-1 14:10	192.168.1.100	1253	56.203.24.91	445	1	48	0	TCP
2014-1-1 14:10	192.168.1.100	1744	163.57.156.13	445	1	48	0	TCP

如果在 Debian 系统中安装了 X-Window 图形环境，那么可以安装 etherape（图形化的网络状况监视器工具）工具，安装方法如下：

```
#apt-get install etherape
```

Etherape 用来监控蠕虫病毒。图 8-6 所示为蠕虫扫描 445 端口的 TCP/IP 连接情况。

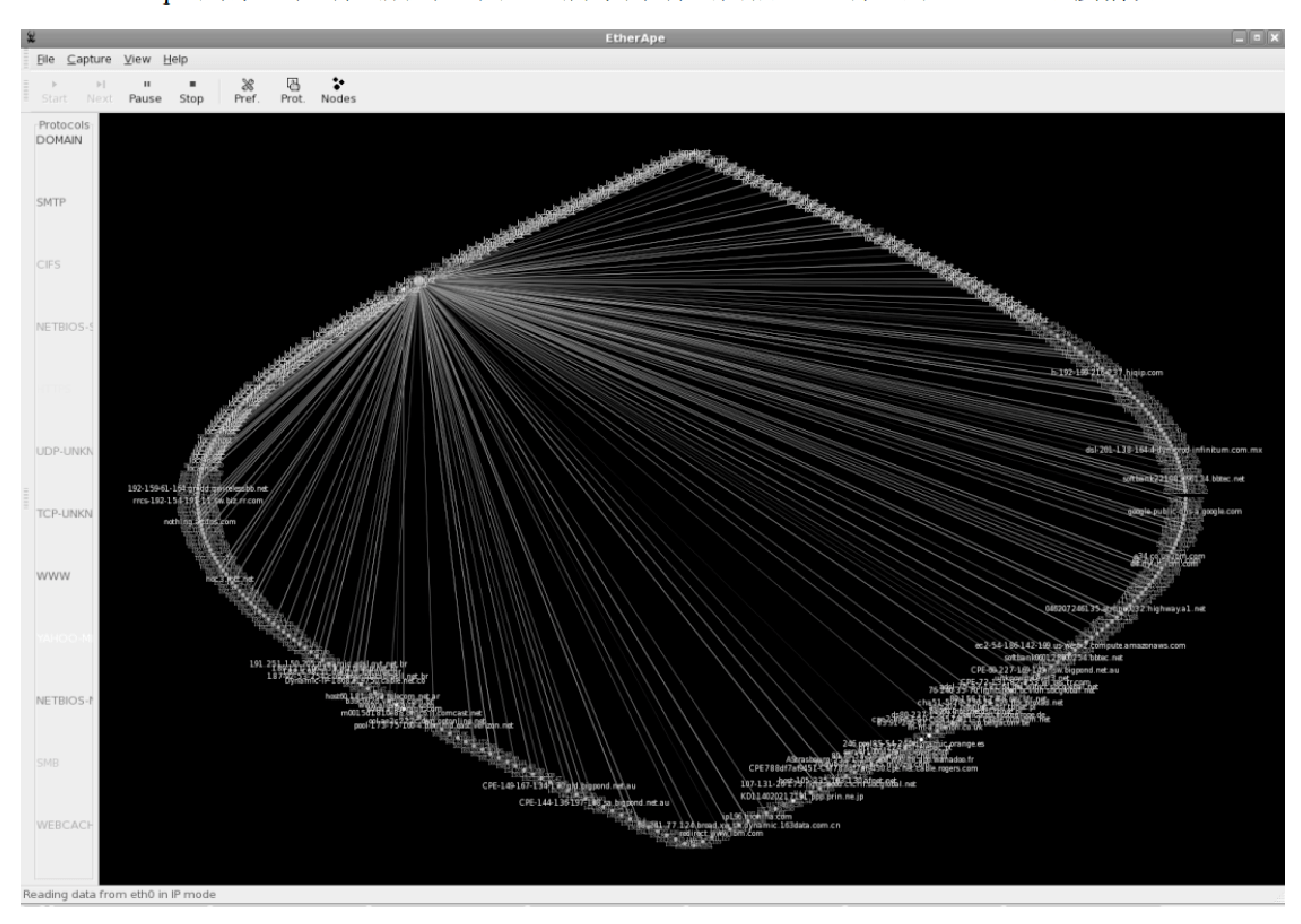


图 8-6　蠕虫对外发起的大量链接

CIFS 消息一般在 Netbios 或 TCP 协议层上，分别使用不同的端口 139 或 445，目前倾向于使用 445 端口。445 是网络邻居使用的端口，如果有计算机连接 445 端口，可能有三种情况：内网的用户偶尔连接，估计是想访问共享文件夹；内网用户频繁连接 445，可以判断对方有病毒。

从表 8-2 中我们可以看出，201.203.1.100 正在对网络 IP 段进行 135 端口扫描。经过查证，这是冲击波（W32.Blaster.Worm）蠕虫。

表 8-2　冲击波（W32.Blaster.Worm）NetFlow 流记录

时间	源 IP	源端口	目的 IP	目的端口	包	字节	标志	协议
2014-1-1 15:10	201.203.1.100	3221	201.203.2.1	135	1	48	0	TCP
2014-1-1 15:10	201.203.1.100	3322	201.203.4.8	135	1	48	0	TCP
2014-1-1 15:10	201.203.1.100	3342	201.203.4.91	135	1	48	0	TCP
2014-1-1 15:10	201.203.1.100	3451	201.203.6.3	135	1	48	0	TCP

例 4：几年前臭名昭著的微软 SQL-Server 漏洞造成了很大的影响，它的特征是目的端口为 1433 的 TCP 流。表 8-3 是根据此条件筛选出的 NetFlow 统计数据，可以看到 IP 地址 66.190.144.166 正在对某网段进行 SQL 漏洞扫描。

表 8-3　筛选的 NetFlow 数据

源 IP	源端口	目的 IP	目的端口	协议	报文数	字节数	B/PK	TOS	Flag
66.190.144.166	6000	202.102.102.33	1433	TCP	1	40	40	00	SYN
66.190.144.166	6000	202.102.102.34	1433	TCP	1	40	40	00	SYN
66.190.144.166	6000	202.102.102.35	1433	TCP	1	40	40	00	SYN
66.190.144.166	6000	202.102.102.34	1433	TCP	1	40	40	00	SYN
66.190.144.166	6000	202.102.102.36	1433	TCP	1	40	40	00	SYN
66.190.144.166	6000	202.102.102.37	1433	TCP	1	40	40	00	SYN

例 5：用 NetFlow 分析 DOS 攻击流量。

DoS 攻击采用大量非正常的数据流量，攻击网络设备或其接入的服务器，致使网络设备的性能下降，或占用网络带宽，影响其他相关用户流量的正常通信。例如 DoS 可以利用 TCP 协议的缺陷，通过 SYN 打开半开的 TCP 连接，占用系统资源，使合法用户被排斥而不能建立正常的 TCP 连接。以下为一个典型的 DoS SYN 攻击的 NetFlow 数据实例，该案例中多个伪造的源 IP 同时向一个目的 IP 发起 TCP SYN 攻击。

```
111.*.68.35|202.*.*.80|Others|64851|3|2|10000|10000|6|1|40|1
105.*.93.91|202.*.*.80|Others|64851|3|2|5557|5928|6|1|40|1
158.*.25.208|202.*.*.80|Others|64851|3|2|3330|10000|6|1|40|1
```

日常工作中发现，除了遇到 DOS 以外，还有许多属于 DDOS 攻击，只不过攻击类别不同，有些是 Ping Death，有些是 SYN flooding。DDOS 攻击基本上都造成这样一种结果：服务器无法处理源源不断的请求，从而造成响应迟缓，直至系统资源耗尽。

因此检测 ICMP 攻击就可以根据下面的条件：在连续的几个时间段，假设每个时间段为 5 分钟，各个时间段内 ICMP 报文大于 5000。符合这个条件的，可以认为受到 ICMP 攻击，或者在用 ICMP 发起攻击，注意协议 01 表示 ICMP。下面是 ICMP 流的 NetFlow 实例。

```
Srcipaddress     dstipaddress    srcp  dstp   sif   dif   proto  pkts  octets
117.234.230.118   67.32.45.33     0     800   0010  0000  01     1989  134920
125.171.109.112   67.32.46.12     0     800   0010  0000  01     1904  122883
112.173.199 122   68.44.34.22     0     800   0010  0000  01     1950  100225
```

另外，还有一种 DOS 攻击是 SYN flooding，我们知道 TCP 协议中有三次握手，如果来源 IP 为伪造，那么三次握手将不能完成，只能停留在 SYN 状态，于是一个 TCP 连接在服务器端就被“挂起”，这种“挂起”将消耗系统资源。大量的伪造源 IP 发起的 SYN 在服务器端被“挂起”，直至系统资源消耗殆尽，这就是 TCP SYN Flooding。它的特征是 TCP 报头中有大量的 SYN 特征数据包。NetFlow 输出格式中提供了 Flag 位，可判断为 SYN 攻击。

因此，检测 SYN flooding 的条件是：在连续的几个时间段，假设每个时间段为 5 分钟，产生大量 flag=2 的数据包，正常连接不会产生这么多 flag=2 的数据包，所以可以设置阈值为 5000。超过该数值就认为服务器受到 SYN flooding 攻击。如果主机发出 flag=2 的数据包数量超过 1000，则可以认为主机在发起攻击，协议号 06 表示 TCP。更多协议号的表示含义可以通过这个页面获得：http://zh.wikipedia.org/wiki/IP（协议号列表）。

以下是 SYN 特征的 NetFlow 实例。

```
Srcipaddress     dstipaddress    srcp  dstp   sif   dif   proto  pkts  octets
117.234.230.118   67.32.45.33     0     800   0010  0000  01     1989  134920
125.171.109.112   67.32.46.12     0     800   0010  0000  01     1904  122883
112.173.199 122   68.44.34.22     0     800   0010  0000  01     1950  100225
```

总之，各种 DDOS 攻击都是在短时间内产生大量的数据包，因此，即使不知道攻击报文的特征，也可以在 NetFlow 的输出结果中进行相应的查找，找到符合条件的异常 Flow。

8.2.4 NetFlow 与谷歌地图的集成显示

在 OSSIM 中 OTX、Ntop 以及资产管理子模块通过利用 Google Map API 实现来访者的 IP 地图定位，同时提供对 IP、域名归属地查询服务，同样在 NetFlow 中也集成了该实用功能。在源 IP 和目标 IP 处均显示了一个我们熟悉的谷歌地图图标，单击后就可以定位该处显示 IP 的位置，如图 8-7 所示。

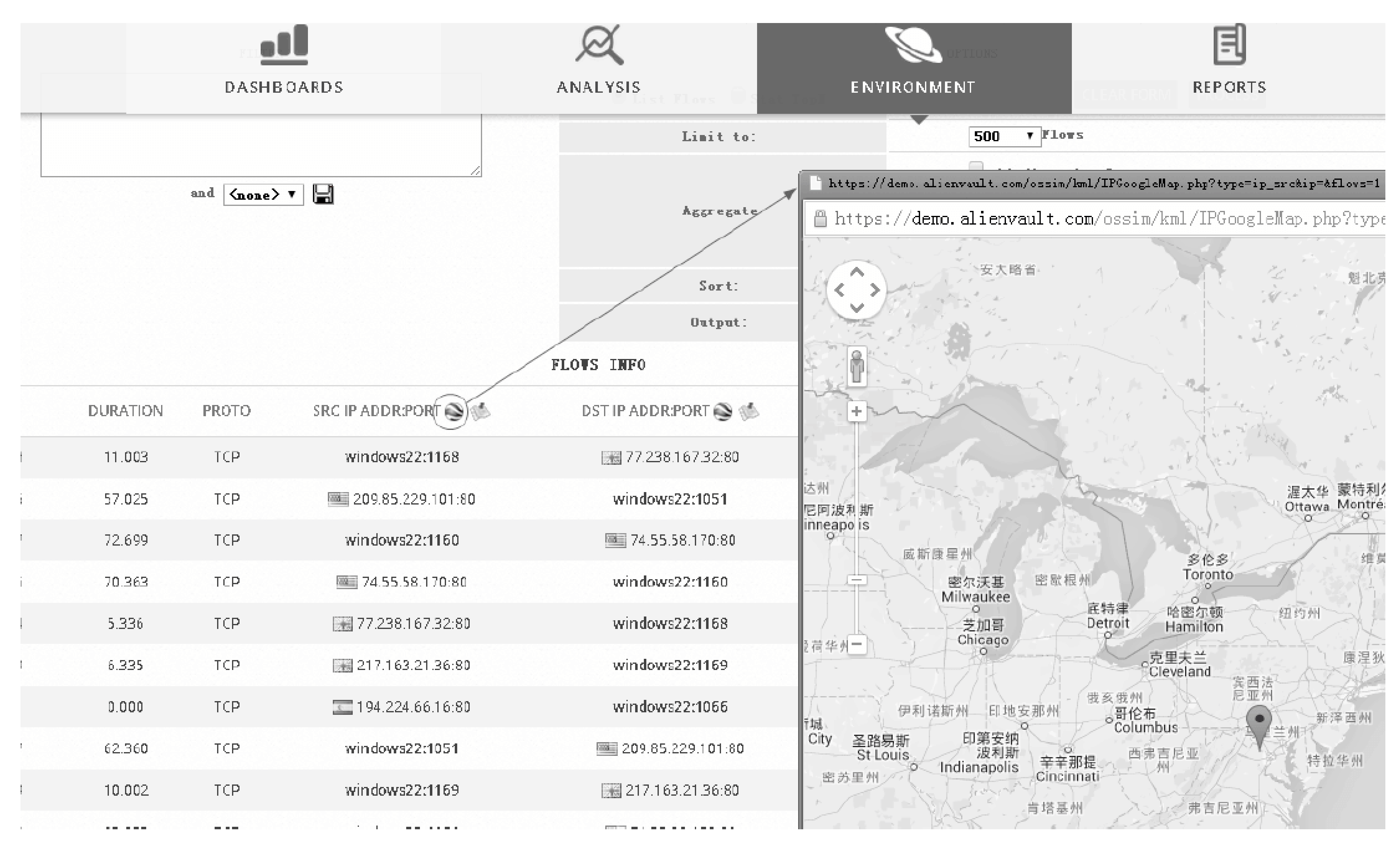


图 8-7　NetFlow 与谷歌地图的集成显示

当地图无法显示时，需要利用 VPN 等方法确保浏览器能连接到谷歌地图。

8.2.5　其他异常流量检测结果分析

私有 IP 地址在公网上是不能被路由的，只能用于局域网内部，当需要与互联网上的其他主机进行通信时，需要使用 NAT 技术将其私有 IP 映射为可以路由的合法 IP。因此，当发现有源 IP 是私有地址的数据包进入企业网，那么可以断定这是为了达到某种目的，而伪造源 IP 的数据流。

例 6：NetFlow 在网络取证方面的应用。

假设图 8-8 中的 ADSL 拨号用户从 Internet 上某 FTP 服务器上下载了可疑文件，在客户端 PC 上留有下载日期时间戳信息，在局端的接入服务器上也可以看到特定 IP 地址在相应时间内被分配给客户端 PC，通过在 ISP 方面的 ANI（Automatic Number Identification）日志就能将客户端的所在家庭电话号码与上网拨号信息联系到一起，与此同时，在 ISP 的路由器上记录（一般会保留 30 天左右）着 FTP 下载/上传网络流量（NetFlow）日志，这个流量至关重要。最后在 FTP 服务器上还有完整的下载记录。

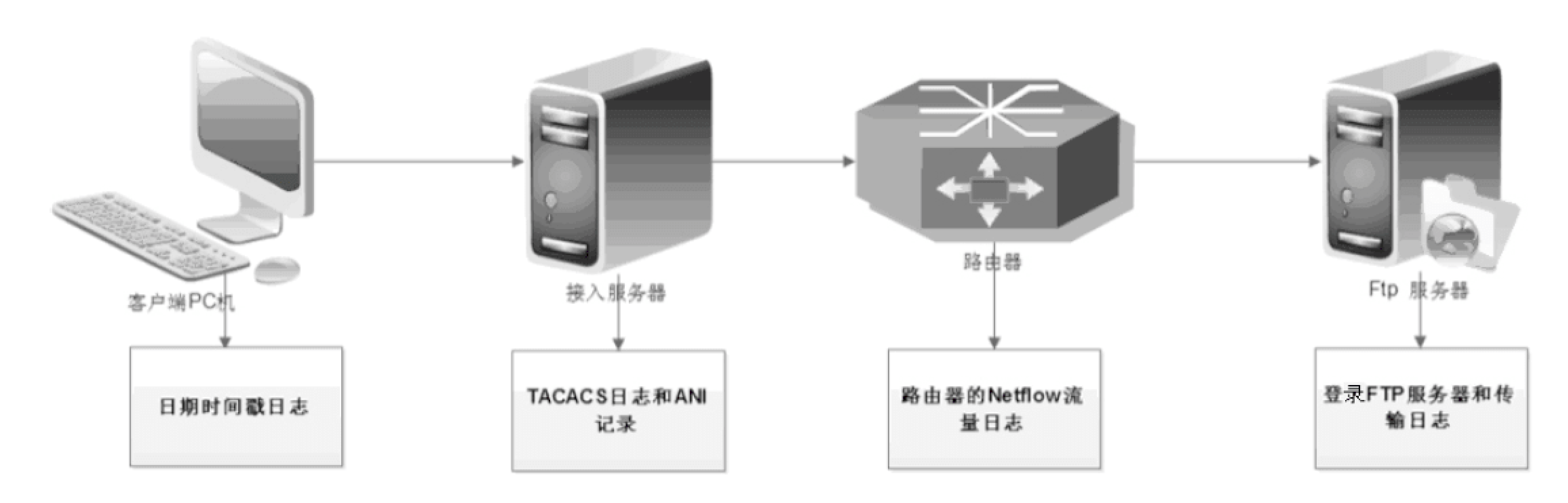


图 8-8 分析下载可疑文件

由上图可以看出，从客户端发起一直到从 FTP 服务器下载分为四个阶段，分别是客户端发送/接收、接入服务器验证、路由器转发及 FTP 服务器接收下载，每个阶段都有日志记录信息包含用户账号、登录时间、IP、端口、发送数据包大小及日期及时间戳等。这些日志信息分别存放在不同的设备上，即便是某些日志遭到了一定程度破坏（例如篡改 IP，丢失了某些日志等）也不会影响全局。

8.3 OSSIM 下 NetFlow 实战

某些情况下，设备不支持 NetFlow，对于这样的环境也有相应的解决方法即使用 Fprobe。如果没有支持 NetFlow 的网络设备，可以利用 fprobe 来生成 NetFlow 报文，其格式为 v5 版本。最初 Fprobe 是一款在 BSD 环境下运行的软件，目前在 UNIX/Linux 平台均可运行。它可以将其接口收到的数据转化为 NetFlow 数据，发送至 NetFlow 分析端。我们可以通过部署 OSSIM 服务器，将网络流量镜像至 OSSIM 服务器以实现对网络流量 NetFlow 分析。在路由器上配置 NetFlow 方法，很多资料都介绍，本节不再赘述。

8.3.1 组成

OSSIM 服务器中的 NetFlow，由下列 3 个工具组成，分别是：

- Fprobe: 从远程 Sensor 主机上发送数据流，在 Sensor 上通过输入“ps –ef|grep fprobe”命令即可查看到通信进程以及端口。
- NfSen: 用于分析图形前端。
- Nfdump: 数据采集模块。

有关 OSSIM 组成结构在第 1 章介绍过，这里先看它是如何分析 NetFlow 数据包的过程，首先在网络接口接收网络数据，然后由 fprobe 程序将收集的数据按照一定规则和格式进行转换为 NetFlow 格式，然后发到系统的 555 端口（通过查看/etc/default/fprobe），然后由 Nfsen 系

统中的 Nfdump 程序将转换后的数据存放在/var/cache/nfdump/flows/live/目录下，最后由 Web 前端程序 Nfsen 读取通过 555 端口接收的数据（可通过查看/etc/ossim/ossim_setup.conf 中的 netflow_remote_collector_port 变量的值得知），反映在前台 Web 界面上（路径 Environment→NetFlow），分析 NetFlow 过程如图 8-9 所示。在 OSSIM 系统中查询 NetFlow 流量如图 8-10 所示。

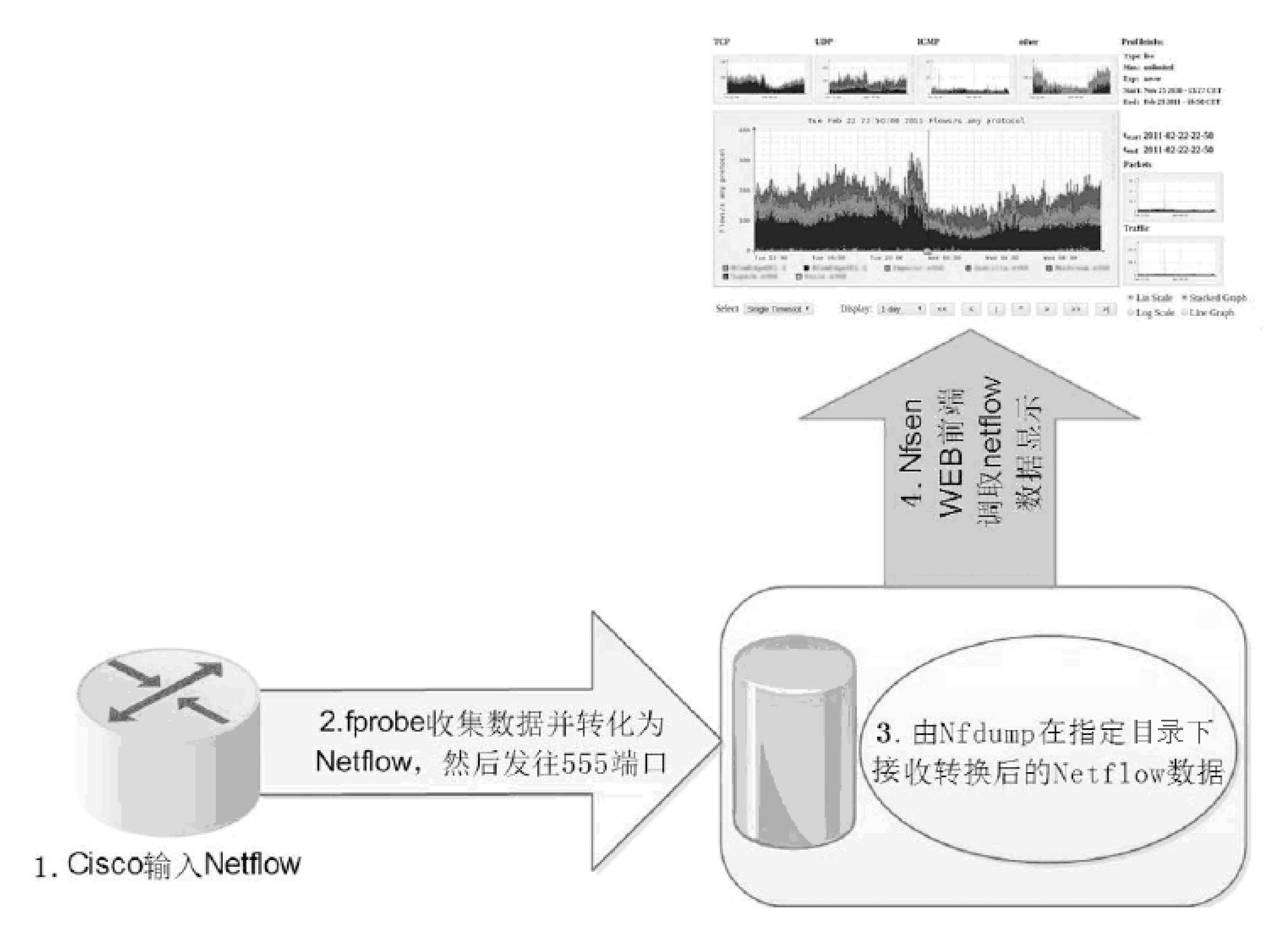


图 8-9　OSSIM 系统分析 NetFlow 数据

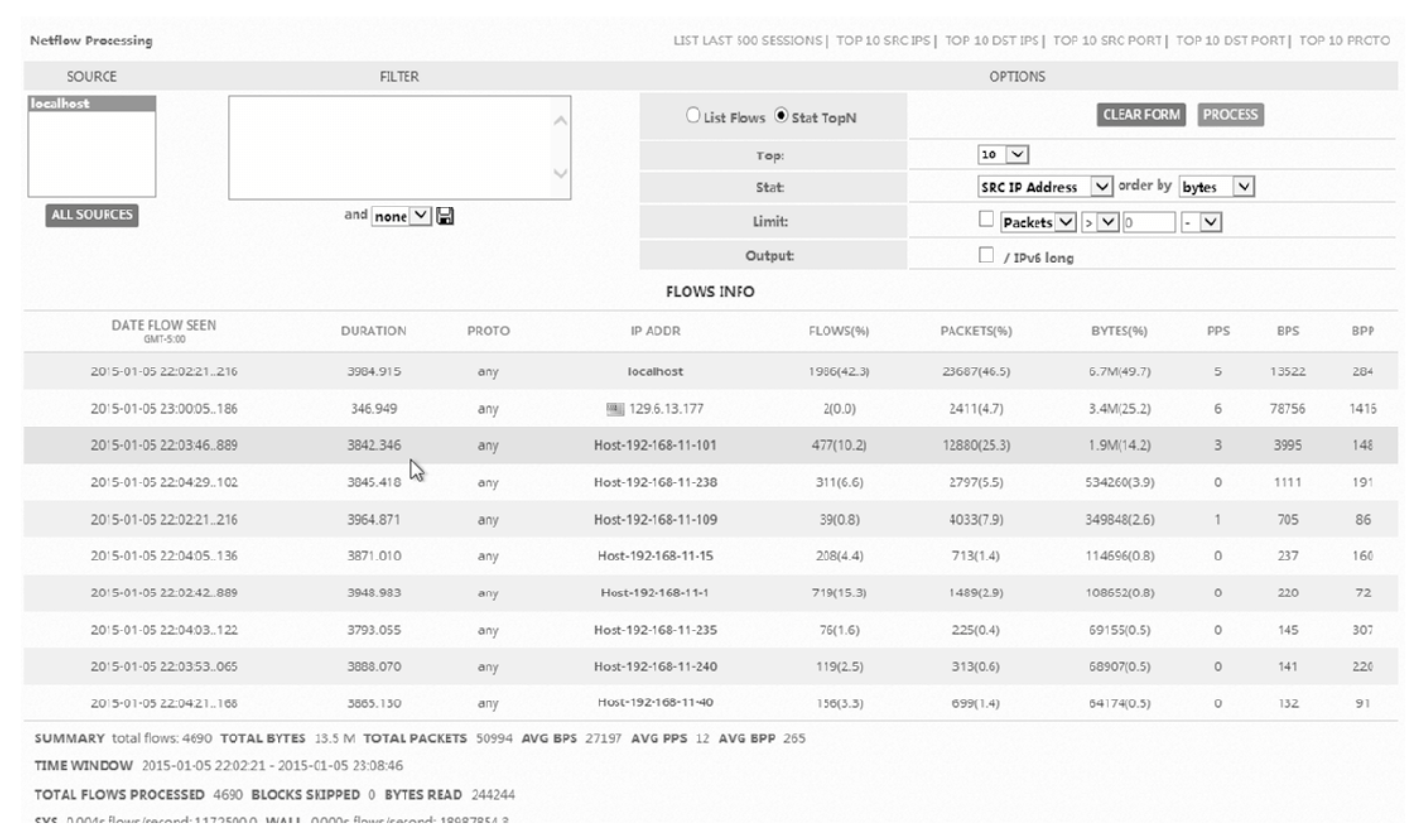
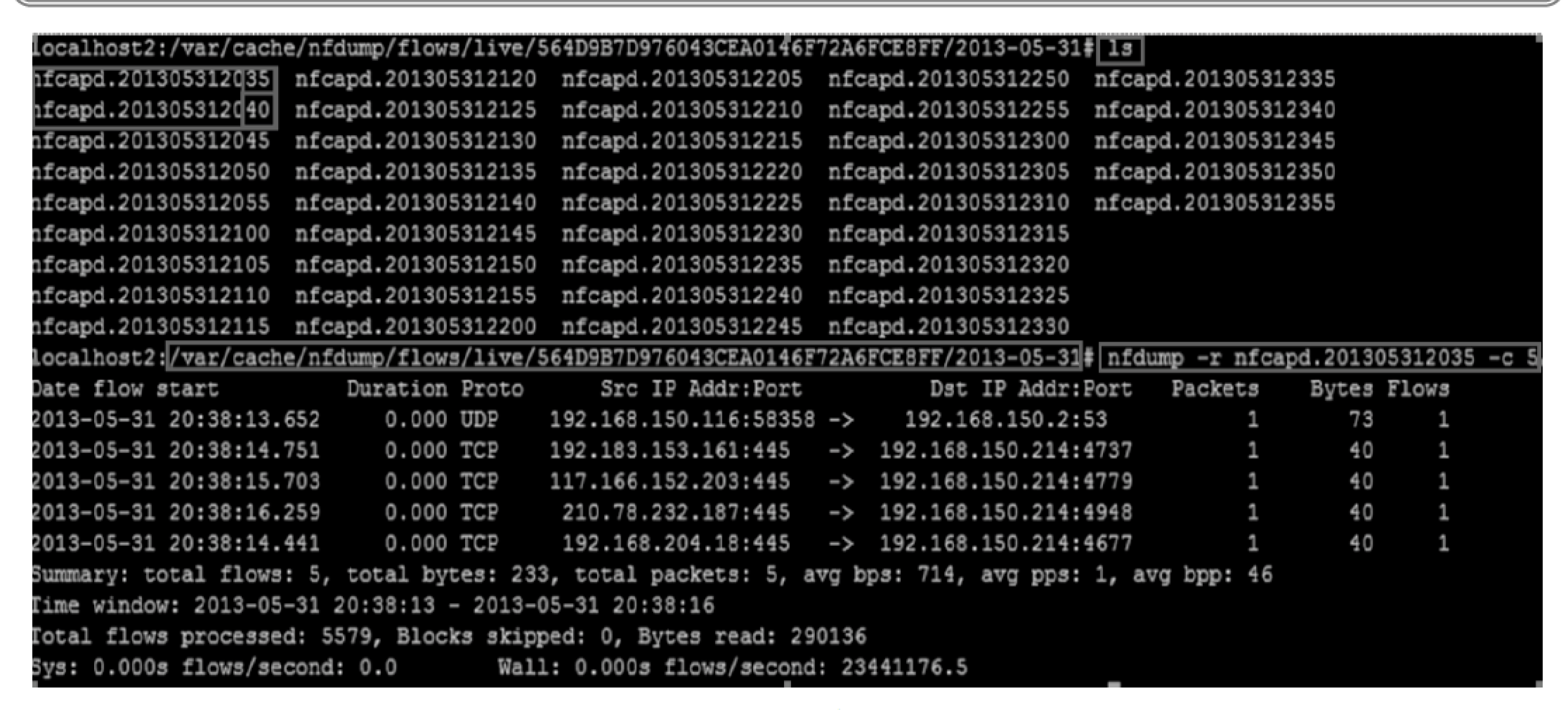


图 8-10　NetFlow 流量查询

NetFlow 数据放置于以日期为单位的/var/cache/nfdump/flows/live/UUID/目录中，存储为二进制文件格式，以天为单位分别设置目录以方便查看，这些数据按照一定的时间组织起来，采集数据间隔为 5 分钟，nfcapd 就建立新的文件，并用时间戳来命名，例如 nfcapd.2013053112035 包含的数据是从 2013 年 5 月 31 日 12 小时 35 分钟开始的数据。如图 8-11 所示。每个目录存放 1 天，总共 288 个文件，每个文件大小从几 KB 到十几 MB 不等，例如：

```
VirtualUSMAllInOne:/var/cache/nfdump/flows/live/564DB4303295CB66AE8A8141C00F6233/2015-05-02# ls
nfcapd.201505020000  nfcapd.201505020215  nfcapd.201505020430  nfcapd.201505020645  nfcapd.201505020900  nfcapd.201505021115
nfcapd.201505020005  nfcapd.201505020220  nfcapd.201505020435  nfcapd.201505020650  nfcapd.201505020905  nfcapd.201505021120
nfcapd.201505020010  nfcapd.201505020225  nfcapd.201505020440  nfcapd.201505020655  nfcapd.201505020910  nfcapd.201505021125
nfcapd.201505020015  nfcapd.201505020230  nfcapd.201505020445  nfcapd.201505020700  nfcapd.201505020915  nfcapd.201505021130
nfcapd.201505020020  nfcapd.201505020235  nfcapd.201505020450  nfcapd.201505020705  nfcapd.201505020920  nfcapd.201505021135
```

以上 UUID 号，每个系统都不相同，可在命令行下通过“alienvault-system-id”查询或者通过查看/etc/ansible/keys 文件获得，两种方法获得结果相同。



```
localhost2:/var/cache/nfdump/flows/live/564D9B7D976043CEA0146F72A6FCE8FF/2013-05-31# ls
nfcapd.201305312035  nfcapd.201305312120  nfcapd.201305312205  nfcapd.201305312250  nfcapd.201305312335
nfcapd.201305312040  nfcapd.201305312125  nfcapd.201305312210  nfcapd.201305312255  nfcapd.201305312340
nfcapd.201305312045  nfcapd.201305312130  nfcapd.201305312215  nfcapd.201305312300  nfcapd.201305312345
nfcapd.201305312050  nfcapd.201305312135  nfcapd.201305312220  nfcapd.201305312305  nfcapd.201305312350
nfcapd.201305312055  nfcapd.201305312140  nfcapd.201305312225  nfcapd.201305312310  nfcapd.201305312355
nfcapd.201305312100  nfcapd.201305312145  nfcapd.201305312230  nfcapd.201305312315
nfcapd.201305312105  nfcapd.201305312150  nfcapd.201305312235  nfcapd.201305312320
nfcapd.201305312110  nfcapd.201305312155  nfcapd.201305312240  nfcapd.201305312325
nfcapd.201305312115  nfcapd.201305312200  nfcapd.201305312245  nfcapd.201305312330
localhost2:/var/cache/nfdump/flows/live/564D9B7D976043CEA0146F72A6FCE8FF/2013-05-31# nfdump -r nfcapd.201305312035 -c 5
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2013-05-31 20:38:13.652     0.000 UDP    192.168.150.116:58358 ->    192.168.150.2:53          1       73     1
2013-05-31 20:38:14.751     0.000 TCP    192.183.153.161:445   -> 192.168.150.214:4737        1       40     1
2013-05-31 20:38:15.703     0.000 TCP    117.166.152.203:445   -> 192.168.150.214:4779        1       40     1
2013-05-31 20:38:16.259     0.000 TCP     210.78.232.187:445   -> 192.168.150.214:4948        1       40     1
2013-05-31 20:38:14.441     0.000 TCP     192.168.204.18:445   -> 192.168.150.214:4677        1       40     1
Summary: total flows: 5, total bytes: 233, total packets: 5, avg bps: 714, avg pps: 1, avg bpp: 46
Time window: 2013-05-31 20:38:13 - 2013-05-31 20:38:16
Total flows processed: 5579, Blocks skipped: 0, Bytes read: 290136
Sys: 0.000s flows/second: 0.0         Wall: 0.000s flows/second: 23441176.5
```

图 8-11　nfcapd 产生的数据

8.3.2　关键参数解释

- “LIST LAST 500 SESSIONS”表示最近 500 个会话列表。
- “TOP 10 SRC IPS”表示源 IP 地址 TOP 10。
- “TOP 10 DST IPS”表示目标 IP 地址 TOP 10。
- “TOP 10 SRC PORT”　表示源端口 TOP 10。
- “TOP 10 DST PORT”表示目标端口 TOP 10。
- “TOP 10 PROTO”表示协议 TOP 10。

TOP 10 表示最多的前 10 位，分布式环境中，部署了多个 Sensor，NetFlow 数据存储在 OSSIM 服务器端/var/cache/nfdump/flows/live/目录下，不同的 Sensor 有不同的 UUID，就用这个 UUID 做为目录来存放来自不同 Sensor 的 NetFlow 数据。

从系统捕获数据包的过程来看，Nfdump 过程至关重要，它包括 nfcpad、fddump、nfprofile 和 nfreplay 这几个进程，功能见表 8-4 所示。

表 8-4　nfdump 工具组成及作用

序号	名 称	作　　用
1	nfcapd 捕获守护进程	从网络中捕获 NetFlow 数据，然后将数据存到文件中。它每隔 n（一般为 5m）分钟在这些文件中轮询一次，必须为每个 NetFlow 流创建一个 nfcapd 进程
2	nfdump 数据挖掘	从由 nfcapd 产生的数据文件中解析出 NetFlow 数据并显示出来，它能够建立大量关于 IP 地址、端口等的 Top N 统计信息，并根据设定顺序显示出来
3	nfprofile 分析器	将 nfcapd 产生的数据文件中解析出 NetFlow 数据，并根据指定的过滤集过滤 NetFlow 数据，并将结果存到文件中
4	nfreplay 数据转发	将 nfcapd 产生的数据文件转发到另一台主机

如果/var/分区空间耗尽，则需要清理空间，首先要考虑清理/var/cache/nfdump/flows/下的文件。

下面我们总结以下实施流量监控步骤：

（1）在 Cisco 6509 上配置 NetFlow（或其他网络设备），并输出到指定到 OSSIM 采集器 IP 的固定 UDP 端口；

（2）采集器软件为 OSSIM 系统的 Flow-tool 工具，该软件监听 UDP 端口，接收进入的 NetFlow 数据包，并存储为特定格式；

（3）使用 Nfsen 软件包中的工具，对 NetFlow 源文件进行读取，并转换成 ASICII 的可读格式，再用 OSSIM 内的 Perl 程序对 NetFlow 进行分析和规范格式等操作，并将读取的 NetFlow 信息存储入 OSSIM 数据库；

（4）依据蠕虫和 DDOS 攻击等异常报文的流量特征，在分析程序中预设各触发条件，定时运行，从中发现满足这些条件的 Flow；

（5）将分析结果在 Web 客户端中展示。

8.3.3　Sensor 中启用 NetFlow

当我们首次将 Sensor 连接到 OSSIM Server 后，默认 NetFlow 功能虽然启用但并没有将数据发送至 NetFlow 采集器，所以我们需要在 Configuration→Deployment→Components→Sensors 下选择对应 Sensor，首次添加 Sensor 时在 Flows 选项中默认 UDP 端口为 12000，显示颜色为蓝色，为了以示区别建议定义其他醒目颜色，如图 8-12 所示。最后单击“CONFIGURE AND RUN”按钮，与此同时在 OSSIM Server 端的/var/cache/nfdump/flows/live 目录下产生一个 UUID 目录，用该目录存储 flow 数据。注意：如果停止了 NetFlow 服务，那么系统就会删除这个目

录以及目录下所有的 flow 数据。在“configuration help”中会给出常见网络设备设置 Netflow 和 sFlow 的详细方法。

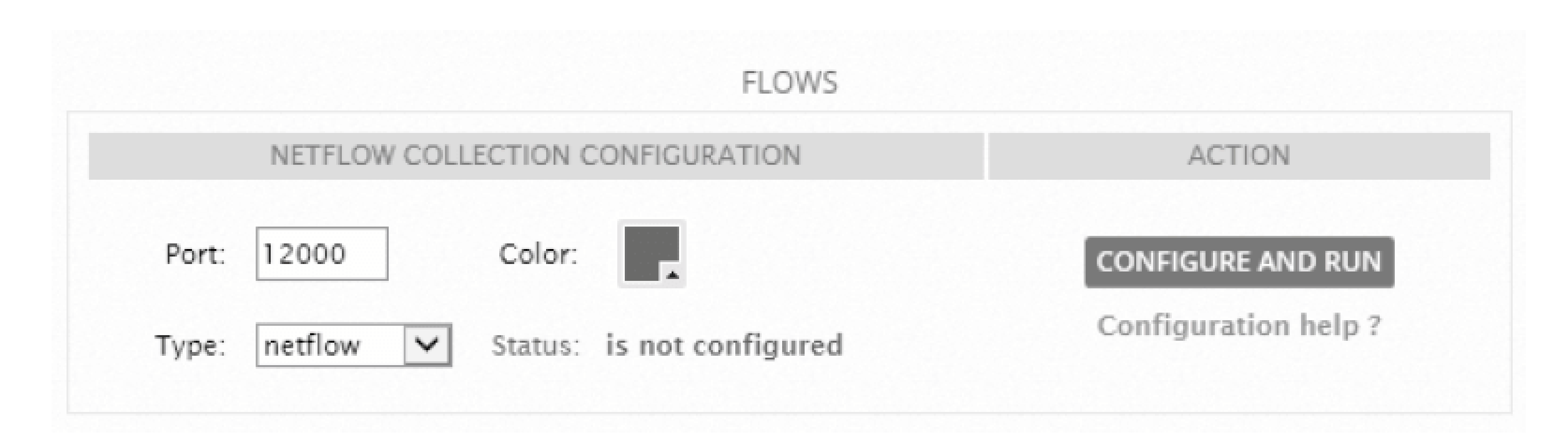


图 8-12　添加 Sensor 的默认 NetFlow UDP 端口为 12000

8.3.4　Nfsen 数据流的存储位置

在 OSSIM 系统中，NetFlow 收集数据流是由/etc/nfsen/nfsen.conf 配置文件中定义，默认路径为/var/cache/nfdump/flows/。由此看出，数据流收集源头同样在该配置文件的%sources 参数中配置。如图 8-13 所示。

```
# Netflow sources
# Define an ident string, port and colour per netflow source
#
# Required parameters:
#    ident   identifies this netflow source. e.g. the router name,
#            Upstream provider name etc.
#    port    nfcapd listens on this port for netflow data for this source
#                        set port to '0' if you do not want a collector to be started
#    col     colour in nfsen graphs for this source
#
# Optional parameters
#    type    Collector type needed for this source. Can be 'netflow' or 'sflow'. Default is netflow
#        optarg  Optional args to the collector at startup
#
# Syntax:
#         'ident' => { 'port' => '<portnum>', 'col' => '<colour>', 'type' => '<type>' }
# Ident strings must be 1 to 19 characters long only, containing characters [a-zA-Z0-9_].

%sources = (
    'ossim'     => { 'port' => '555', 'col' => '#0000ff', 'type' => 'netflow' }
,    'alienvault'     => { 'port' => '12000', 'col' => '#ff0048', 'type' => 'netflow' }
);
```

图 8-13　NetFlow 中数据源配置

另外，当在 Sensor 上启用 NetFlow 后，在 Server 上的 iptables 规则会自动添加一条规则允许 sensor 将收集到的流发往 Server 端 UDP 12000 端口，读者可以在 Server 端通过命令检查。

```
# iptables -L |grep 12000
```

若添加第二个 Sensor，并启用 NetFlow，则端口号为 12001，以此类推每个 Sensor 的 NetFlow 端口号不能重复。

这是 OSSIM 2.3 系统中的配置，在 OSSIM 4 系统中把主机名换成了类似“676A3FF2D6834353970D95DE0”的一串 32 位的 UUID 号（字母数字的组合），它保证收集数据流的网络接口不会重复。例如：/etc/nfsen/nfsen.conf 配置文件如图 8-14 所示。

```
# Required parameters:
#    ident   identifies this netflow source. e.g. the router name,
#            Upstream provider name etc.
#    port    nfcapd listens on this port for netflow data for this source
#                        set port to '0' if you do not want a collector to be started
#    col     colour in nfsen graphs for this source
#
# Optional parameters
#    type    Collector type needed for this source. Can be 'netflow' or 'sflow'. Default is netflow
#        optarg  Optional args to the collector at startup
#
# Syntax:
#         'ident' => { 'port' => '<portnum>', 'col' => '<colour>', 'type' => '<type>' }
# Ident strings must be 1 to 19 characters long only, containing characters [a-zA-Z0-9_].

%sources = (
    '289B628009C944AB889D9AE874C40F90'    => { 'port' => '555', 'col' => '#0000ff', 'type' => 'netflow'
}
,    '38C4F098C5DA41E4A8C4BF83000FD944'    => { 'port' => '12000', 'col' => '#ff2600', 'type' => 'netflo
w' }
);
```

图 8-14　/etc/nfsen/nfsen.conf 配置文件

如果修改了 nfsen.conf 配置文件，要使其生效需执行以下命令：

```
#nfsen reconfig
```

重启 nfsen 服务：

```
#/etc/init.d/nfsen restart
```

下面我们查看 UUID 号在数据库中的位置，如图 8-15 所示。

```
alienvault:/var/cache/nfdump/flows/live# ls
289B628009C944AB889D9AE874C40F90  38C4F098C5DA41E4A8C4BF83000FD944
alienvault:/var/cache/nfdump/flows/live# ossim-db
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14634
Server version: 5.5.33-31.1 Percona Server (GPL), Release 31.1

Copyright (c) 2009-2013 Percona LLC and/or its affiliates
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE alienvault;
Database changed
mysql> SELECT hex(id),name FROM sensor;
+----------------------------------+-------------+
| hex(id)                          | name        |
+----------------------------------+-------------+
| 4327ADFBBBC347809795B2107E53D218 | sensor      |
| 61A65634045211E4B31808002789E94E | alienvault  |
+----------------------------------+-------------+
2 rows in set (0.00 sec)

mysql> 
```

图 8-15　查询 UUID

在分布式 OSSIM 环境中，用不同颜色表示多个 Sensor。如图 8-16 所示。

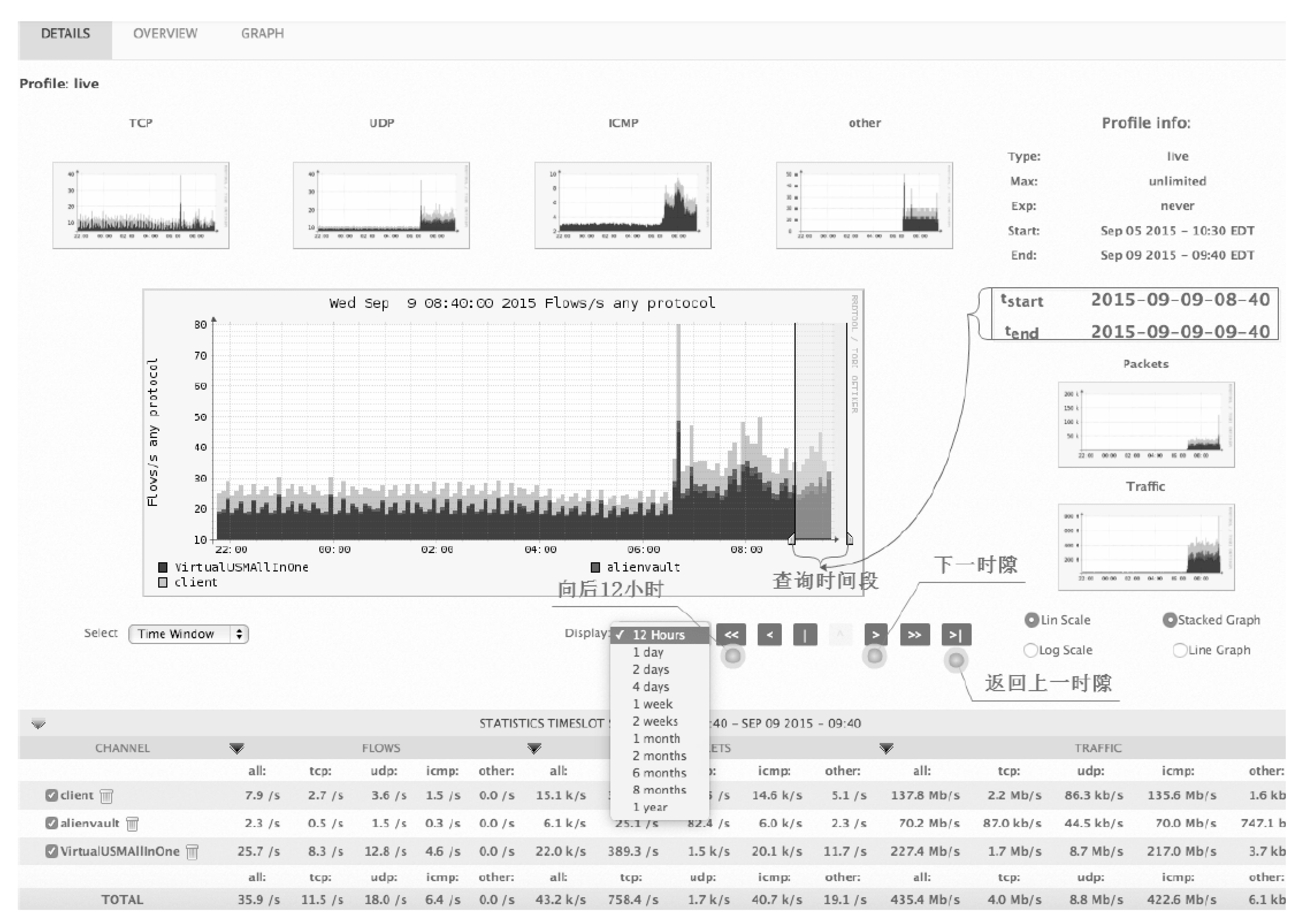


图 8-16　多个 Sensor 之间用不同颜色区分流量大小

停止 NetFlow 服务后，才可更改显示颜色。

8.3.5　NetFlows 抽样数据保存时间

NetFlow 每隔 5 分钟保存一次数据包，所以需要定期移走这些包，它们保存时间不能过长，系统默认保存 45 天。实际应用中大家可以根据自己磁盘的大小设置，调整的具体路径为 Configuration→Administration→Main→Backup，0 代表永久存储，这里我们修改为 15 天。在这里有关存储的事件条数的限制，默认为 400 万条。

8.3.6　NetFlow 的读取方式

读取 NetFlow 数据即可通过命令行方式，也可图形化方式展现，作为用户当然是喜欢后者，但命令行读取方式同样重要。

1. 命令行方式

命令行方式读取 NetFlow 数据，如图 8-17 所示。

```
alienvault:/var/cache/nfdump/flows/live/564DED48907BBD8BFCCA5E63C74B47E6/2015-01-16# nfdump  -r nfcapd.201501160050
Date flow start          Duration Proto      Src IP Addr:Port           Dst IP Addr:Port   Packets    Bytes Flows
2015-01-16 00:49:31.553    27.442 TCP       192.168.91.1:61593 ->   192.168.91.129:443          11     1724     1
2015-01-16 00:49:47.294     0.004 UDP       192.168.91.2:53    ->   192.168.91.129:59421         2      228     1
2015-01-16 00:48:48.289    43.248 TCP     192.168.91.129:443   ->     192.168.91.1:61526         8     1078     1
2015-01-16 00:48:48.290    43.246 TCP       192.168.91.1:61531 ->   192.168.91.129:443          15     2539     1
2015-01-16 00:48:48.289    43.248 TCP       192.168.91.1:61527 ->   192.168.91.129:443          13     1873     1
2015-01-16 00:49:31.555    27.440 TCP       192.168.91.1:61595 ->   192.168.91.129:443          11     1708     1
2015-01-16 00:48:48.290    43.246 TCP     192.168.91.129:443   ->     192.168.91.1:61530        10     1355     1
2015-01-16 00:48:48.289    43.248 TCP       192.168.91.1:61528 ->   192.168.91.129:443          15     2587     1
2015-01-16 00:48:48.289    43.248 TCP     192.168.91.129:443   ->     192.168.91.1:61527        10     1371     1
2015-01-16 00:48:48.289    43.248 TCP     192.168.91.129:443   ->     192.168.91.1:61528        12     1632     1
2015-01-16 00:48:48.289    43.248 TCP     192.168.91.129:443   ->     192.168.91.1:61529        12     1616     1
2015-01-16 00:49:31.551    27.445 TCP     192.168.91.129:443   ->     192.168.91.1:61591         8     1864     1
2015-01-16 00:49:47.293     0.000 UDP     192.168.91.129:59421 ->     192.168.91.2:53            2      120     1
2015-01-16 00:49:31.551    27.445 TCP       192.168.91.1:61591 ->   192.168.91.129:443          11     1756     1
2015-01-16 00:48:48.289    43.248 TCP       192.168.91.1:61526 ->   192.168.91.129:443          11     1191     1
2015-01-16 00:48:48.290    43.246 TCP       192.168.91.1:61530 ->   192.168.91.129:443          13     1873     1
2015-01-16 00:49:31.554    27.448 TCP       192.168.91.1:61594 ->   192.168.91.129:443          11     1724     1
2015-01-16 00:48:47.986    43.550 TCP     192.168.91.129:443   ->     192.168.91.1:61523        11     3618     1
2015-01-16 00:49:31.550    27.445 TCP     192.168.91.129:443   ->     192.168.91.1:61590         8     1768     1
2015-01-16 00:49:47.300     0.000 TCP     192.168.91.129:44423 ->  173.194.127.144:80            1       52     1
2015-01-16 00:49:31.552    27.453 TCP     192.168.91.129:443   ->     192.168.91.1:61592         8     1768     1
2015-01-16 00:49:55.310     0.000 TCP     192.168.91.129:58094 ->  173.194.127.148:80            1       52     1
2015-01-16 00:48:47.986    43.550 TCP       192.168.91.1:61523 ->   192.168.91.129:443          14     1737     1
2015-01-16 00:48:48.290    43.246 TCP     192.168.91.129:443   ->     192.168.91.1:61531        12     1632     1
```

图 8-17　命令行方式读取 NetFlow 数据

以上命令读出 NetFlow 采样。查看采样结果，这里对目标地址进行汇总，默认取 Top 10，并按默认的流排序/flows，也可以按字节/bytes 排序，如图 8-18 所示。

```
alienvault:/var/cache/nfdump/flows/live/564DED48907BBD8BFCCA5E63C74B47E6/2015-01-16# nfdump  -r nfcapd.201501160050 -s dstip/flo
ws
Top 10 Dst IP Addr ordered by flows:
Date first seen          Duration Proto       Dst IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps      bps   bpp
2015-01-16 00:48:47.986   291.063 any      192.168.91.129    57(49.6)      598(56.2)     92711(52.4)         2     2548   155
2015-01-16 00:48:47.986   264.468 any        192.168.91.1    49(42.6)      431(40.5)     72080(40.8)         1     2180   167
2015-01-16 00:49:47.293   229.283 any        192.168.91.2     3( 2.6)        4( 0.4)       246( 0.1)         0        8    61
2015-01-16 00:53:36.577     2.472 any         64.62.160.45     1( 0.9)       27( 2.5)     11568( 6.5)        10    37436   428
2015-01-16 00:49:51.307     0.000 any     173.194.127.146     1( 0.9)        1( 0.1)        52( 0.0)         0        0    52
2015-01-16 00:49:55.310     0.000 any     173.194.127.148     1( 0.9)        1( 0.1)        52( 0.0)         0        0    52
2015-01-16 00:49:49.303     0.000 any     173.194.127.145     1( 0.9)        1( 0.1)        52( 0.0)         0        0    52
2015-01-16 00:49:53.308     0.000 any     173.194.127.147     1( 0.9)        1( 0.1)        52( 0.0)         0        0    52
2015-01-16 00:49:47.300     0.000 any     173.194.127.144     1( 0.9)        1( 0.1)        52( 0.0)         0        0    52

Summary: total flows: 115, total bytes: 176865, total packets: 1065, avg bps: 4861, avg pps: 3, avg bpp: 166
Time window: 2015-01-16 00:48:47 - 2015-01-16 00:53:39
Total flows processed: 115, Blocks skipped: 0, Bytes read: 6008
Sys: 0.004s flows/second: 28750.0    Wall: 0.000s flows/second: 1000000.0
alienvault:/var/cache/nfdump/flows/live/564DED48907BBD8BFCCA5E63C74B47E6/2015-01-16#
```

图 8-18　读取 Top10

2. 图形化方式

下面介绍图形化方式典型架构：Fprobe+nfsen 的流量分析。在这种架构中，首先安装 Linux 开发环境、fprobe 及 libpcap-devel，然后让 fprobe 监听 eth0 数据并输出 NetFlow 到某个端口，例如：

```
# fprobe -i eth0 127.0.0.1:9995
```

测试是否收到 NetFlow 数据：

```
#tcpdump -i lo port 9995    \\lo 为环路接口名称。
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 65535 bytes
10:36:53.000221 IP localhost.40448 > localhost.palace-4: UDP, length 552
10:37:03.000222 IP localhost.40448 > localhost.palace-4: UDP, length 792
10:37:13.000215 IP localhost.40448 > localhost.palace-4: UDP, length 1224
10:37:23.000220 IP localhost.40448 > localhost.palace-4: UDP, length 792
… …
```

此时，表示已经成功地将端口镜像的数据转化为 NetFlow 数据，并发送至本机的 UDP 9995 端口，接着安装 nfdump 和 nfsen。

8.3.7　nfdump 的作用

nfdump 是一款开源的 NetFlow 收集、存储、过滤及统计分析软件。Nfsen 是基于 nfdump 的 Web 工具，所以服务器需要先安装 LAMP 环境。当然，其他组件如 RRD Tool、Perl 模块也必须安装。接着安装 nfdump 工具和 nfsen。

最后修改 nfsen 配置文件，并重启 nfsen 进程。将 NetFlow 数据流发送到 nfsen 配置好的端口，即可浏览 Neflow 数据。这几个步骤看着虽然不复杂，但读者亲自实战时需要费些功夫，各种报错问题时常出现。在 Debian Linux 中手动安装 nfdump 的效果整体上和 OSSIM 中的 NetFlow 比起来，功能仍然相对单一。

8.3.8　将 NetFlow 数据集成到 Web UI 的仪表盘

出于用户观察数据需要，常需要将 NetFlow 的历史数据，在首页仪表盘中调用。下面介绍设置方法，系统默认在 Dashboards 内没有启用 Network 监控。大家首先进入 Overview 菜单，单击右侧签字笔状图标 ，会立刻出现如图 8-19 所示的界面。

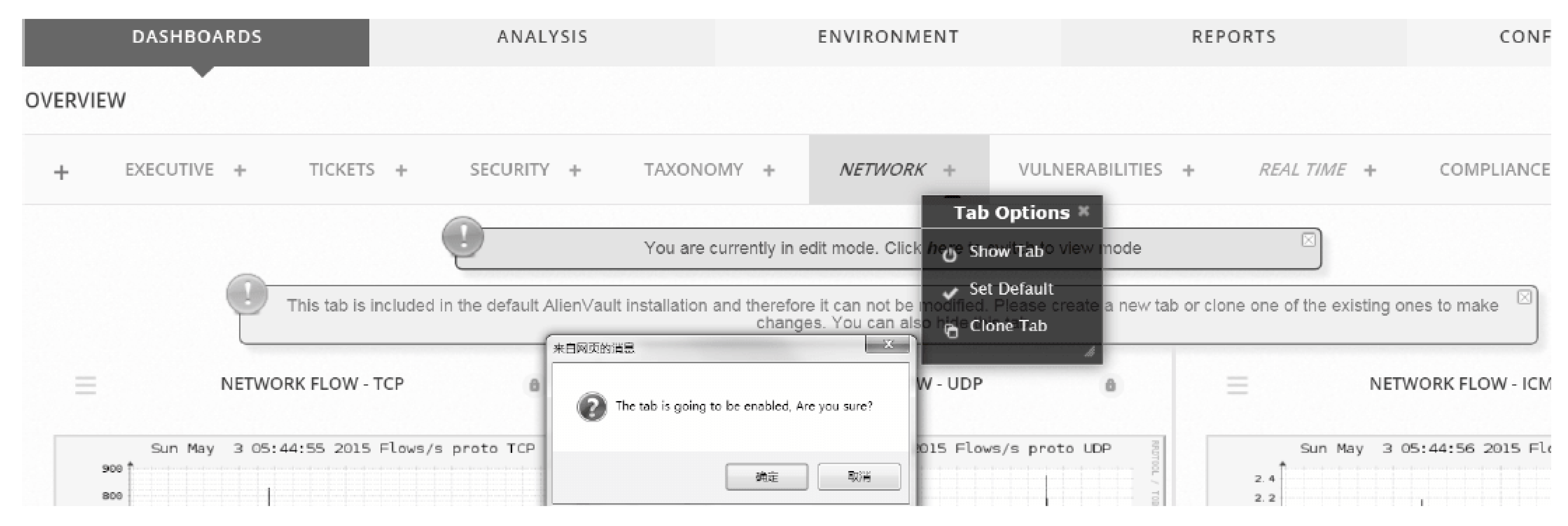


图 8-19　添加到仪表盘

在图 8-19 中单击斜体字的 Network 按钮，并选择“Show Tab”选项，在弹出对话框中选择确定按钮。设置完成后，便立即在 Web UI 中看到 OVERVIEW 多出了个 Network 按钮，单击此按钮即可预览 NetFlow 历史数据，如图 8-20 所示。

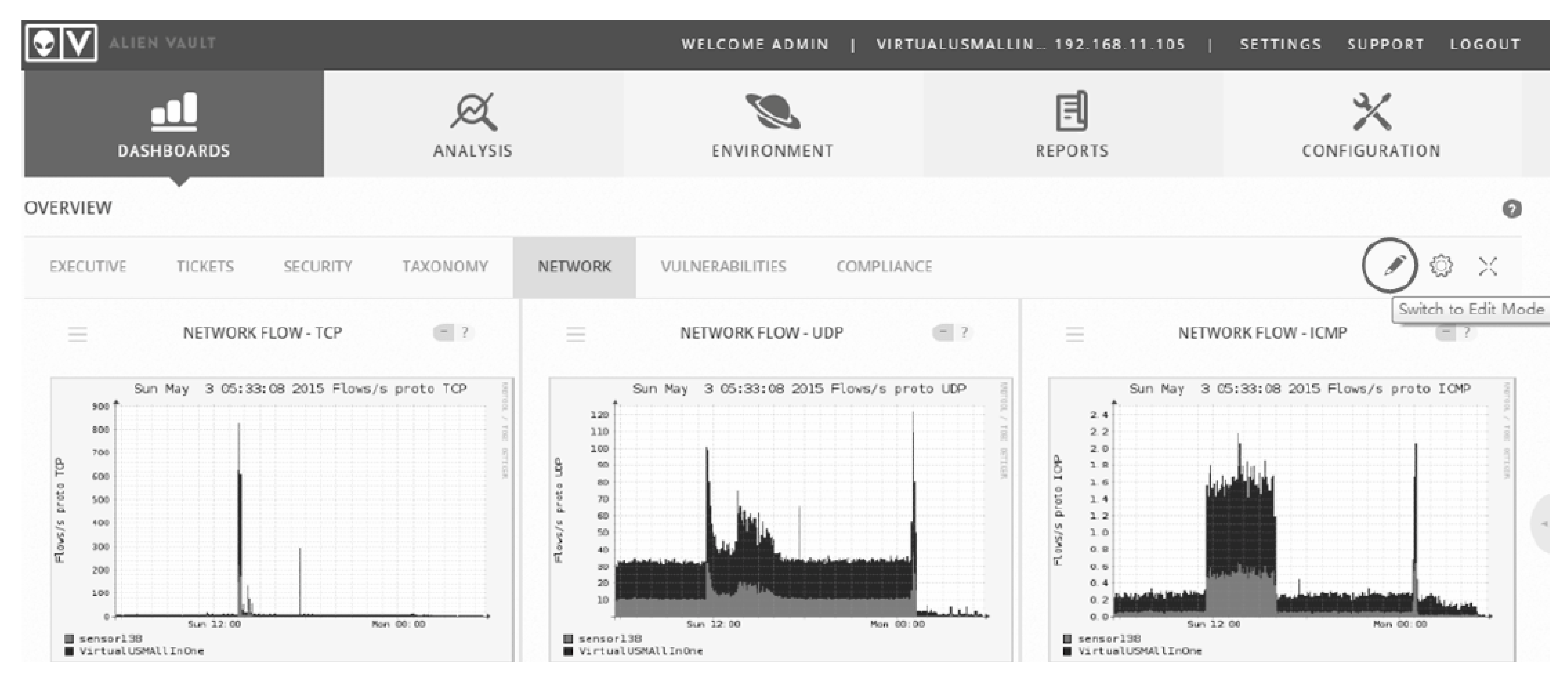


图 8-20　在仪表盘中显示 NetFlow

8.3.9　分布式环境下 NetFlow 数据流处理

本小节内容是对上述知识点的总结，下面这个实验在一个模拟的分布式环境中完成，其中有一台混合安装的 OSSIM USM，两台 Sensor，三台接入层交换机，一台核心交换机以及若干 PC 组成，为简化实验，这些设备均布署在同一个 VLAN 中，实际生产中 Sensor 应分布在多个 VLAN。拓扑如图 8-21 所示。

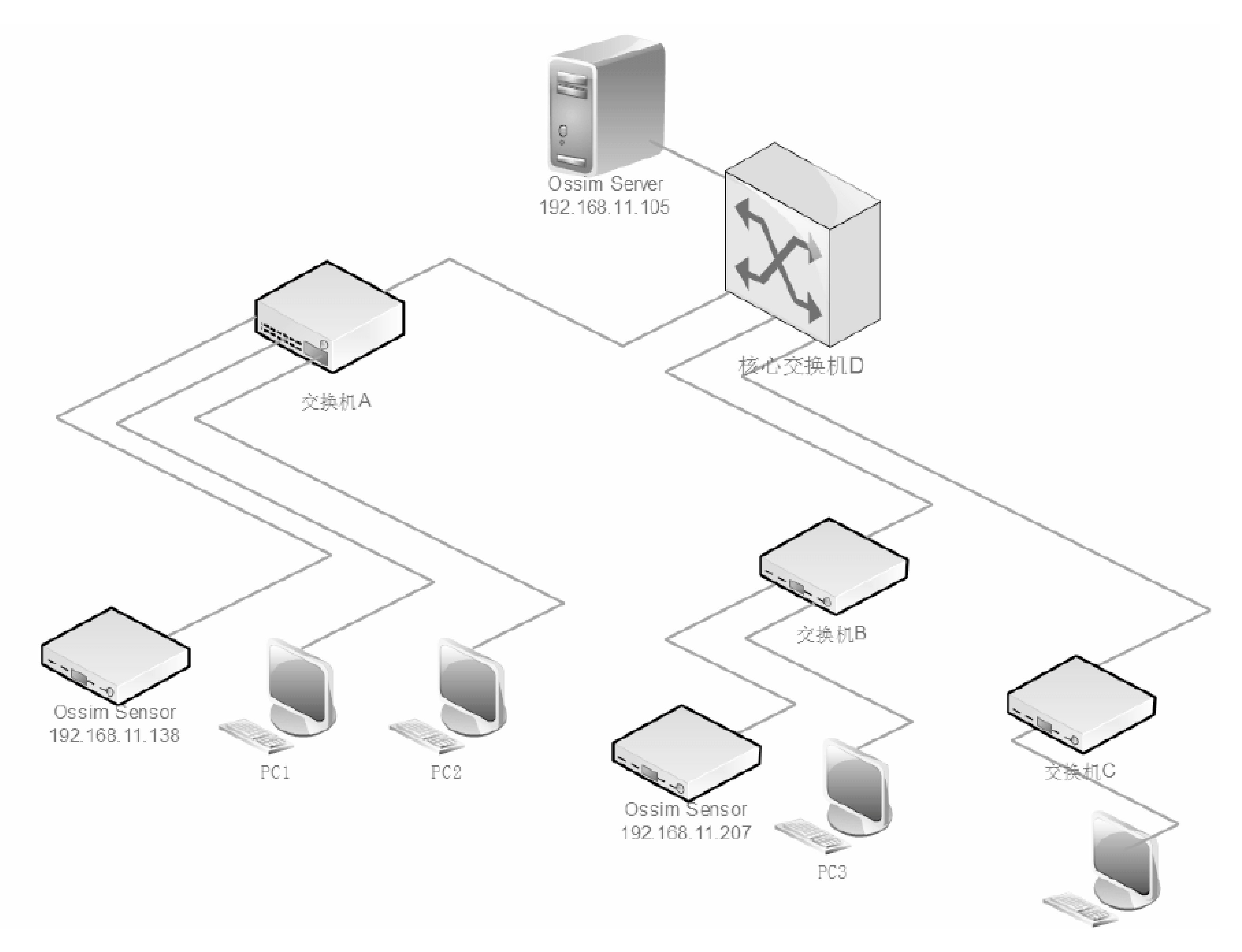


图 8-21　OSSIM　分布式部署

第 2 章讲解了如何添加 Sensor，并与 Server 进行连接，下面接着启用 NetFlow 服务，交换机上 NetFlow 也必须同时设置正确，注意不同 Sensor 主机中的 nfcapd 进程分别在 12000 和 120001 端口进行监听。各主机 IP、UUID 及端口如表 8-5 所示。注意 Sensor 是集成在完全安装的 OSSIM USM 中，即 192.168.11.105 的传感器，SensorA 和 SensorB 都只配了一块网卡。

表 8-5　Ossim 服务器传感器配置

主机名	IP	UUID	NetFlow 端口	安装方式
SensorA	192.168.11.138	cd8a4e60-bb14-4e4e-99a0-3700d8ec73aa	12000	Sensor
SensorB	192.168.11.207	73e61ef5-d3c5-4168-a8fe-5561bf5e903b	12001	Sensor
Sensor	192.168.11.105	564db430-3295-cb66-ae8a-8141c00f6233	555	USM 完全安装

前面已经讲过查看 UUID 的方法，下面依次在终端下输入如下命令：

对于 Sensor 192.168.11.105，操作如下：

```
VirtualUSMAllInOne:~# cat /etc/alienvault/system-id
564db430-3295-cb66-ae8a-8141c00f6233
```

对于 SensorA 192.168.11.138，操作如下：

```
alienvault:~# cat /etc/alienvault/system-id
cd8a4e60-bb14-4e4e-99a0-3700d8ec73aa
```

对于 SensorB 192.168.11.207，操作如下：

```
alienvault:~# cat /etc/alienvault/system-id
73e61ef5-d3c5-4168-a8fe-5561bf5e903b
```

所有数据存储在 Ossim Server 端，接下来最重要的步骤需要将 SensorA、SensorB 采样数据流转发到 192.168.11.105 主机上，这一点在 Web UI 上并没有直接的界面，操作如下：

在 SensorA 上操作:

```
#/usr/sbin/fprobe -i eth0 -fip 192.168.11.105:12000
```

在 SensorB 上操作：

```
#/usr/sbin/fprobe -i eth0 -fip 192.168.11.105:12001
```

大家在实验室，不可完全照搬命令，192.168.11.105 主机后面的端口 12000、12001 是在 Sensor 上启用 NetFlow 服务时随机分配的，需要读者将 Sensor IP 地址和这个随机分配的端口号的对应关系记清楚。

OSSIM Server 端是否收到这些数据呢？需要用如下命令进行验证。

```
#tcpdump -n udp port 12000
#tcpdump -n udp port 12001
```

如果系统重启，那么这条命令又需要重新输入，这是我们可以分别修改 SensorA、B 上的

ossim_setup.conf 配置文件。

在 SensorA 上操作：

```
#vi /etc/ossim/ossim_setup.conf
```

修改第 55 行 NetFlow_remote_collector_port=555，在此行配置中将 555 默认端口修改为对应的 netflow 发送端口，以 SensorA 为例该值为 12000。

为了使其生效不必重启系统，只要执行以下命令：

```
#ossim-reconfig
```

在 SensorB 上操作：

```
#vi /etc/ossim/ossim_setup.conf
```

修改第 55 行：

```
netflow_remote_collector_port=12001
```

运行以下命令：

```
#ossim-reconfig
```

这时，系统会自动修改/etc/default/fprobe 配置文件中 Flow_collector 的端口号。

为了确保在 SensorA、SensorB 和 Sensor 上都能展现其 NetFlow 数据，要确保 nfsen 都收到数据，我们在 Ossim Server 的终端控制台下，操作以下命令：

```
VirtualUSMAllInOne:~# nfsen -r live
name    live
group   (nogroup)
tcreate Tue Mar 27 11:55:00 2012
tstart  Fri Apr 10 07:25:00 2015
tend    Thu Apr 30 16:20:00 2015
updated Thu Apr 30 16:20:00 2015
expire  0 hours
size    339.9 MB
maxsize 0
type    live
locked  0
status  OK
version 130
channel 564DB4303295CB66AE8A8141C00F6233        sign: + colour: #0000ff order: 1        sourcelist: 564DB430
3295CB66AE8A8141C00F6233        Files: 1937     size: 355409920
channel CD8A4E60BB144E4E99A03700D8EC73AA        sign: + colour: #ff00ee order: 2        sourcelist: CD8A4E60
BB144E4E99A03700D8EC73AA        Files: 155      size: 634880
channel 73E61EF5D3C54168A8FE5561BF5E903B        sign: + colour: #8ff24e order: 3        sourcelist: 73E61EF5
D3C54168A8FE5561BF5E903B        Files: 82       size: 335872
```

在输出结果中，“size”的值代表 SensorA、SensorB 和 Sensor 三个传感器发送采样数据流的总容量。

接着观察目录的内容，我们进入目录 564db4303295cb66ae8a8141c00f6233，在该目录下有若干 Pcap 格式的文件，每隔 5 分钟生成一个，每个文件大小在 500KB~1000KB（大小并不固定），每天会产生 100~200MB 的抓包文件。

```
VirtualUSMAllInOne:/var/cache/nfdump/flows/live# ls -l
total 12
drwxrwxr-x 3 www-data www-data 4096 Apr 30 17:05 564DB4303295CB66AE8A8141C00F6233
drwxrwxr-x 3 www-data www-data 4096 Apr 30 17:05 73E61EF5D3C54168A8FE5561BF5E903B
drwxrwxr-x 3 www-data www-data 4096 Apr 30 17:05 CD8A4E60BB144E4E99A03700D8EC73AA
VirtualUSMAllInOne:/var/cache/nfdump/flows/live#
```

如果在实验中发现没有收到 Sensor 数据包，首先检查设置是否正确，接下来用 tcpdump 来抓包分析，具体抓包操作如下所示。

```
VirtualUSMAllInOne:~# ps aux  |grep fprobe
root      6051  0.0  0.0   6028   704 pts/4    S+   17:26   0:00 grep --color=auto fprobe
root     15420  0.4  0.0  47336  6456 ?        Ssl  16:00   0:25 /usr/sbin/fprobe -ieth0 -fip 192.168.11.105:555
VirtualUSMAllInOne:~# tcpdump -i eth0 -n 'host 192.168.11.105 and port 555'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:26:34.553754 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 120
17:26:36.301592 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 120
17:26:46.299573 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 264
17:26:49.485444 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 888
17:26:59.486335 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 504
17:27:01.518599 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 72
17:27:09.470212 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 264
17:27:24.496926 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 216
17:27:26.282860 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 1464
17:27:29.520040 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 936
17:27:29.520090 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 1464
17:27:31.280089 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 1464
17:27:34.563715 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 120
17:27:39.560335 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 216
17:27:44.499525 IP 192.168.11.138.49799 > 192.168.11.105.555: UDP, length 600
17:27:46.275407 IP 192.168.11.207.51498 > 192.168.11.105.555: UDP, length 984
```

或 tcpdump -i eth0 ‘port 12001’ 。

通过 nfdump 可以实现 NetFlow 记录的过滤、Top 统计和排序等功能，在 OSSIM 通过 Web UI 能轻松地展现给用户，如图 8-22 所示。

图 8-22　NetFlow 多传感器显示

作为系统维护人员需要了解后台执行的命令，例如我们进入 2015-05-01 目录，对其数据包按端口排序输入命令“nfdump –r nfcapd.201505010315 –s dstport –n 10”。

```
VirtualUSMAllInOne:/var/cache/nfdump/flows/live/564DB4303295CB66AE8A8141C00F6233/2015-05-01# nfdump -r nfcapd.201505010315 -s dstport -n 10
Top 10 Dst Port ordered by flows:
Date first seen          Duration Proto         Dst Port   Flows(%)     Packets(%)       Bytes(%)         pps      bps   bpp
2015-05-01 03:16:34.453 236739.118 any               53   498(24.1)      770( 4.8)    54061( 1.3)        0        1    70
2015-05-01 03:16:20.473    154.140 any             4672   268(13.0)      284( 1.8)     9245( 0.2)        1      479    32
2015-05-01 03:16:30.936 236742.944 any               80    99( 4.8)      389( 2.4)    48593( 1.1)        0        1   124
2015-05-01 03:16:14.920 236759.314 any               22    83( 4.0)     2634(16.4)    2.1 M(48.2)        0       70   790
2015-05-01 03:17:23.666 236688.567 any             8000    42( 2.0)       43( 0.3)     2730( 0.1)        0        0    63
2015-05-01 03:16:56.976 236715.252 any               21    40( 1.9)       41( 0.3)     2430( 0.1)        0        0    59
2015-05-01 03:17:26.666 236688.557 any             8080    40( 1.9)       56( 0.3)     3360( 0.1)        0        0    60
2015-05-01 03:17:15.954 236699.934 any             5355    17( 0.8)       34( 0.2)     1700( 0.0)        0        0    50
2015-05-01 03:16:40.752    119.666 any            15000    11( 0.5)       13( 0.1)      390( 0.0)        0       26    30
2015-05-01 03:17:35.517 236701.942 any              768    10( 0.5)      268( 1.7)    26112( 0.6)        0        0    97

Summary: total flows: 2068, total bytes: 4.3 M, total packets: 16014, avg bps: 145, avg pps: 0, avg bpp: 269
Time window: 2015-05-01 03:13:50 - 2015-05-03 21:02:51
Total flows processed: 2068, Blocks skipped: 0, Bytes read: 107564
Sys: 0.000s flows/second: 0.0        Wall: 0.001s flows/second: 1973282.4
VirtualUSMAllInOne:/var/cache/nfdump/flows/live/564DB4303295CB66AE8A8141C00F6233/2015-05-01#
```

又如：

```
#nfdump -R ./ -s dstport -n 10
#nfdump -r nfcapd.201505011610 -n 10 -s proto
```

8.4 OSSIM 流量监控工具综合应用

网络管理中除了要找出网络性能的瓶颈、信息安全防护，还应该掌握网络带宽的具体使用情况，从网络流量的变化，能够发现异常行为。当某 PC 感染病毒时，会出现比平常高出许多倍的流量，此时封锁该 IP 的联网，才能阻止病毒继续蔓延。

通过长期流量监控，可以使我们能建立网络评价基线标准，某个时段对主机通信流量的观察，可以掌握主机在不同时段的流量值，一旦收集到足够多的数据，就可建立起基线标准。本节重点讨论如何使用 OSSIM 下的 Ntop 和 Nagios 这两款流量监控工具。

8.4.1　Ntop 流量采集方式

Ntop 与一些基于 SNMP 来获取网络设备的端口流量不同，Ntop 可以分析二、三层及更高层流量。Ntop 还能根据所监测网络特征，灵活改变流量采集方法，可以更加精准地采集网络流量。

（1）当监测一个网段的流量状况，Ntop 可采用基于 Sniffer 的流量采集模式，可以说 Ntop 也是一种网络嗅探器。嗅探器在协助监控网络数据传输、排除网络故障等方面有着其他工具所不可替代的作用。

例如，发现疑似网络攻击行为，通过嗅探器截获的数据包可以确定正在攻击系统的是什么类型的数据包以及它们的源头，从而可以及时做出响应，这些 Ntop 独有的特性是 Cacti 以及 Zabbix 等监控工具所无法提供的。

（2）当监测 Cisco 路由器交换机（它们支持 NetFlow 流）时，可以通过对设备配置打开采集网络数据流功能，通过 UDP 协议，用 NetFlow 接收流量数据，然后将收集的数据归档。

（3）H3C、HP ProCurve 2900、Foundry 设备都支持 sFlow 收集网络流量。由于 sFlow 被

部署在交换机和路由器的 ASIC 中，所以属于基于硬件的流量采集技术。通过 sFlow 对网络进行监控将不需要镜像监控端口，这样就节约了网络资源消耗。Ntop 这里就相当于 sFlow 收集器，可以接收 sFlow agent 发来的数据流。

sFlow 传输协议也是 UDP，端口为 6343。

Ntop 需要对捕获的数据包进行实时、快速分类，Ntop 的哈希算法采用了散列算法的基本思想，Ntop 的数据包分析器在某个时间单元内分析数据包，它根据已用的网络接口对数据包的头部信息进行分析。而各个主机的信息被存放在一个包含计数器的巨大 Hash 表中，这些计数器是根据相应网络协议生成，它们通过排序号的主机跟踪数据的收发。由于我们不能预知被控主机的数量，Ntop 有可能占用相当数量的内存，因此 Ntop 被设计为定时清空主机列表，以防止可用内存资源被耗尽，这样设计还防止哈希表过于庞大。

8.4.2 Ntop 监控

在 OSSIM 系统中默认集成了 Ntop 工具，Web 浏览器方式查看 Ntop，启动方式为 Environment→Profiles, 打开的界面如图 8-23 所示。

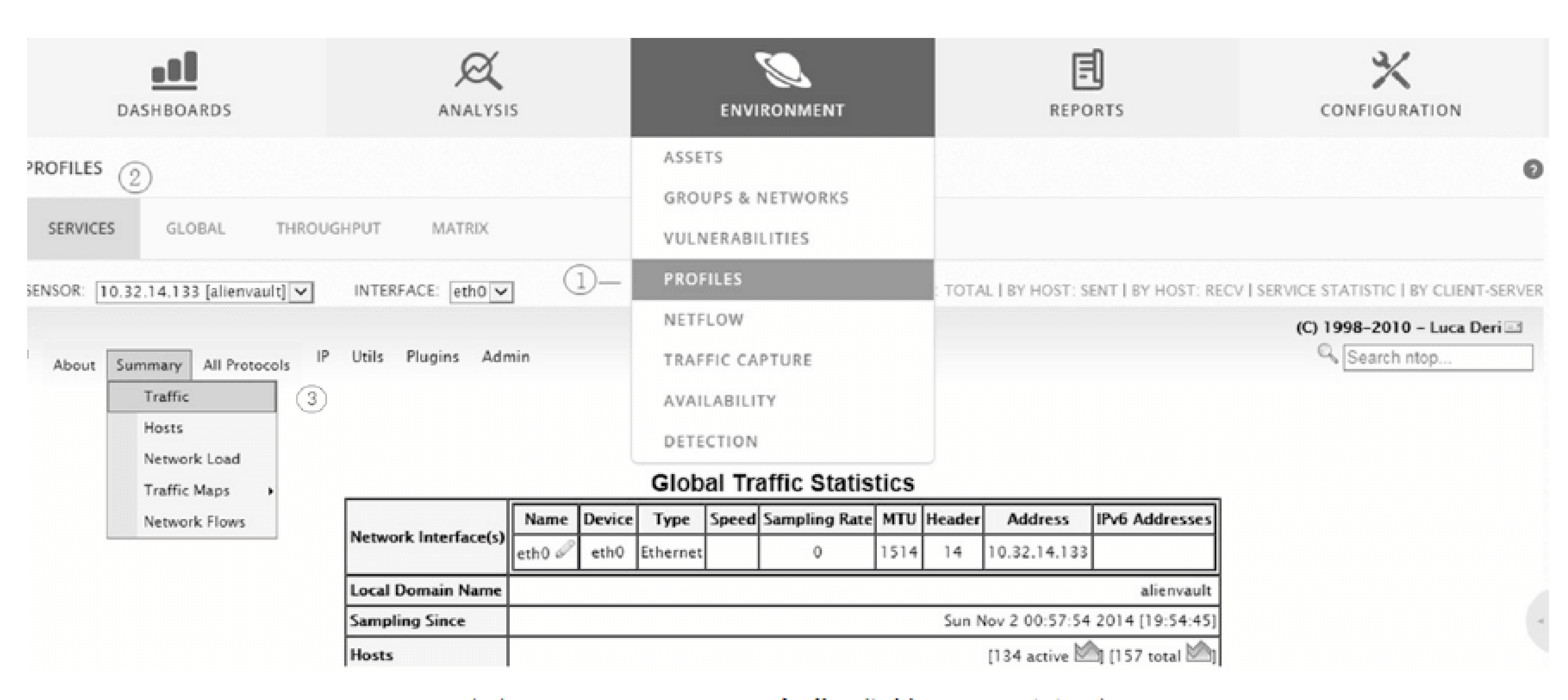


图 8-23　OSSIM 中集成的 Ntop 界面

用 NTOP 进行网络流量分析查询的过程中，网络管理员可以查看各种数据统计界面，来对流量进行分析，前提是必须在交换机上做好 SPAN 设置。Ntop 网络流量分析功能如表 8-6 所示。

表 8-6　网络流量分析功能菜单说明

功能菜单	子菜单	功　能
Summary	Traffic	显示网卡探测的所有流量
	Hosts	显示主机信息，分别以字节、报文为单位显示，同时又可以基于 VLAN 查看不同的主机信息。而且每一列都支持排序，方便用户快速查询所需信息

（续表）

功能菜单	子菜单	功　能
Summary	Network Load	显示的被监控接口的网络负载
	Traffic map	显示网络流量图（细分为区域地图和主机地图）
	Network Flows	显示 Host Last Seen 插件和用户自定义的规则情况
All Protocols	Traffic	包含所有协议的网络流量，以及所有主机的发送和接收报文情况
	Throughput	包含所有协议的网络负载情况，包括所有主机的发送和接收报文情况
	Activity	包含所有协议的网络活动情况
IP	Summary 统计各应用层协议信息	Traffic，能显示本网络主机和远程主机发送和接收数据的信息
		Multicast，查看组播统计信息
		Internet Domains，显示互联网域统计信息
		Networks，显示当前监控网段信息
		Ass，显示自治域信息
		Distribution，显示本地、远程到本地、远程的流量和协议分布
	Traffic Directions 记录内网和外网各个流向的流量统计	Local to local，本地主机流量
		Local to remote，本地到远程主机的 IP 流量
		Remote to local，远程到本地的主机 IP 流量
		Remote to remote，远程主机之间的 IP 流量
	Local，记录本地的端口号，TCP 连接以及主机操作系统信息	Routes，本地子网路由信息
		Ports Used，本地主机的服务端口使用情况
		Active TCP/UDP Sessions，活动 TCP/UDP 会话
		Host Fingerprints，主机指纹
		Hosts Characterization，主机特征
		Network Traffic Map，本地网络中主机间的连接图
		Traffic Matrix，监控本地子网内主机之间的流量，以表单形式给出统计结果。网络管理员可以根据这些信息从整体上监控本地子网内主机之间的数据交互情况
Utils	RRD Alarm	RRD 配置
	Data Dump	导出数据，提供把网络中各主机的流量通过各种文档格式导出，以供后续查询使用

Ntop 本地主机特征子项及含义如表 8-7 所示。

表 8-7　Local Hosts Characterization（本地主机特征）

子　项	含　义
Host	主机信息。一般为主机的 IP 地址（也可以是 DNS 名称或者 NetBios 等），为蓝色超链接形式，可以通过[Summary/Hosts]菜单项对应页面，查看该主机记录，单击此主机弹出新的页面显示具体流量统计等信息
Unhealthy Host	服务器端的主机名称或 IP 地址

（续表）

子　项	含　义
L2 Switch Bridge	对应主机是否为 2 层设备
Gateway	对应主机是否为网关
VoIP Host	是否为 VoIP 主机
Printer	对应主机是否为打印机
NTP/DNS Server	对应主机是否为 NTP/DNS Server
SMTP/POP/IMAP Server	对应主机是否为邮件服务器
Directory/FTP/HTTP Server	对应主机是否为 Directory/FTP/HTTP Server
DHCP/WINS Server	对应主机是否为 DHCP/WINS Server
DHCP Client	对应主机是否为 DHCP Client
P2P	是否为 P2P 服务器
Total	以上每种服务器的数目

Ntop 还可以识别本地主机在网络中提供的服务和其他一些特征，从而判断这些主机在网络中担当的角色。如图 8-24 所示，角色还包括存在安全隐患的主机（包括 MAC 地址冲突、使用的端口、连接的主机连超过 1024 等情况），图中列出了每种图标的含义。

低风险　DNS
中风险　HTTP Server
VOIP　P2P Server

Local Hosts Characterization

Host	Unhealthy Host	L2 Switch Bridge	Gateway	VoIP Host	Printer	NTP/DNS Server	SMTP/POP/IMAP Server	Directory/FTP/HTTP Server	DHCP/WINS Server	DHCP Client	P2P
10.32.14.252	X										
mp1piozzjoodv9l [NetBIOS]	X										
10.32.14.131	X			X		X					X
alienvault.alienvault	X							X			X
Total	4 [50.0 %]			1		1		1			0

图 8-24　本地主机的角色

另外还有一些重要功能，例如查看本地子网路由信息，如图 8-25 所示。查看本地各主机特征，如图 8-26 所示。查看 Netfows Flows 信息（前提为在 Ntop 中设置了 NetFlow），如图 8-27 所示。查看自治区的信息，如图 8-28 所示。列出流量最高的主机，如图 8-29 所示。查看激活端口，如图 8-30 所示。查看本地协议使用情况，如图 8-31 所示。

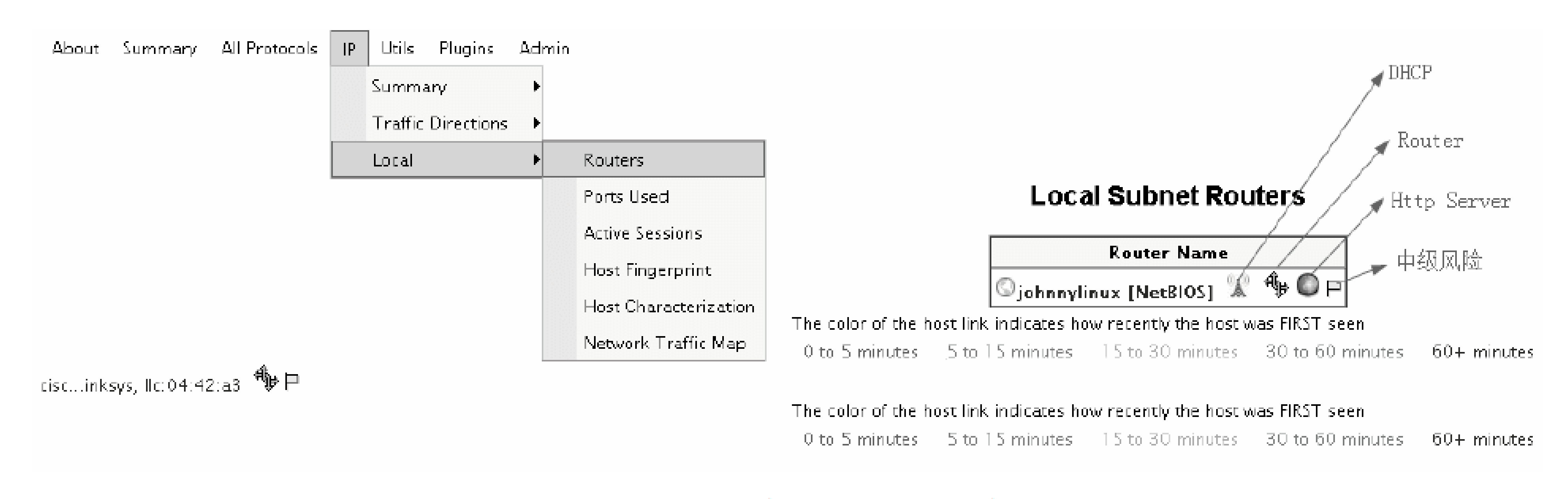


图 8-25　查看本地子网路由信息

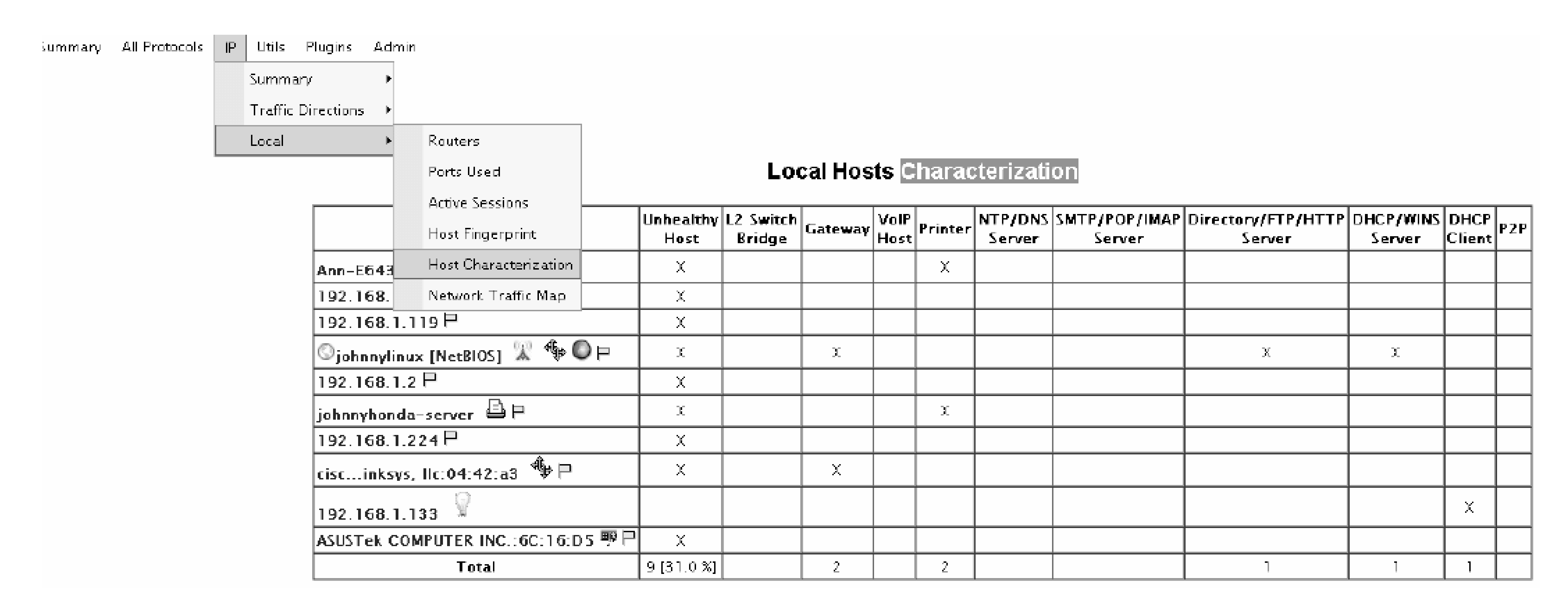


图 8-26　查看本地各主机特征

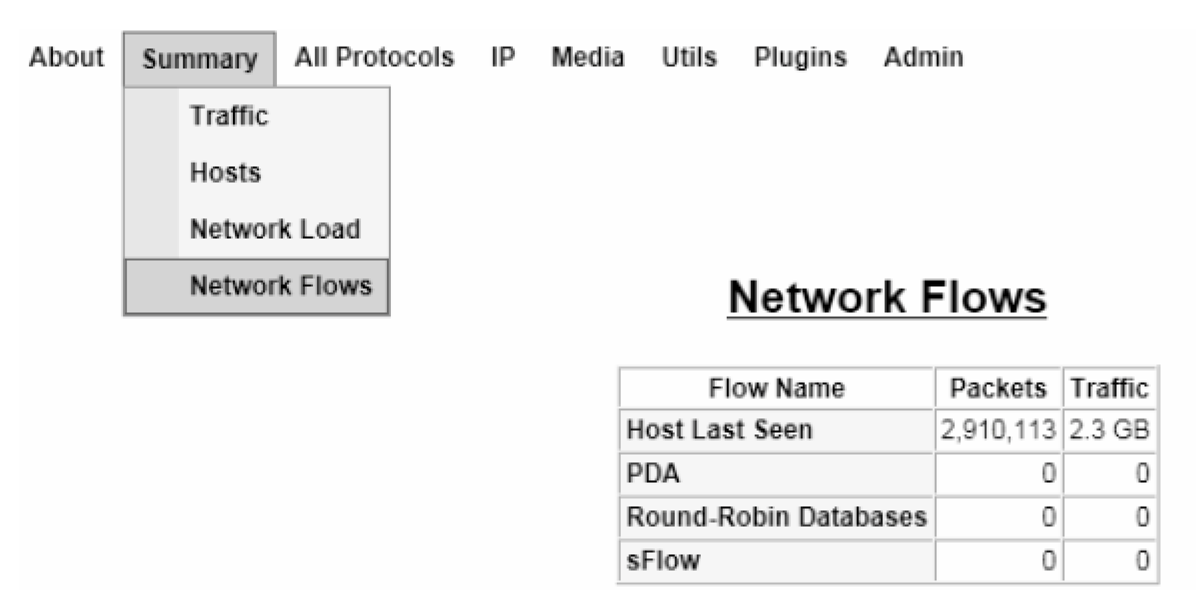


图 8-27　查看 Netfows Flows 信息

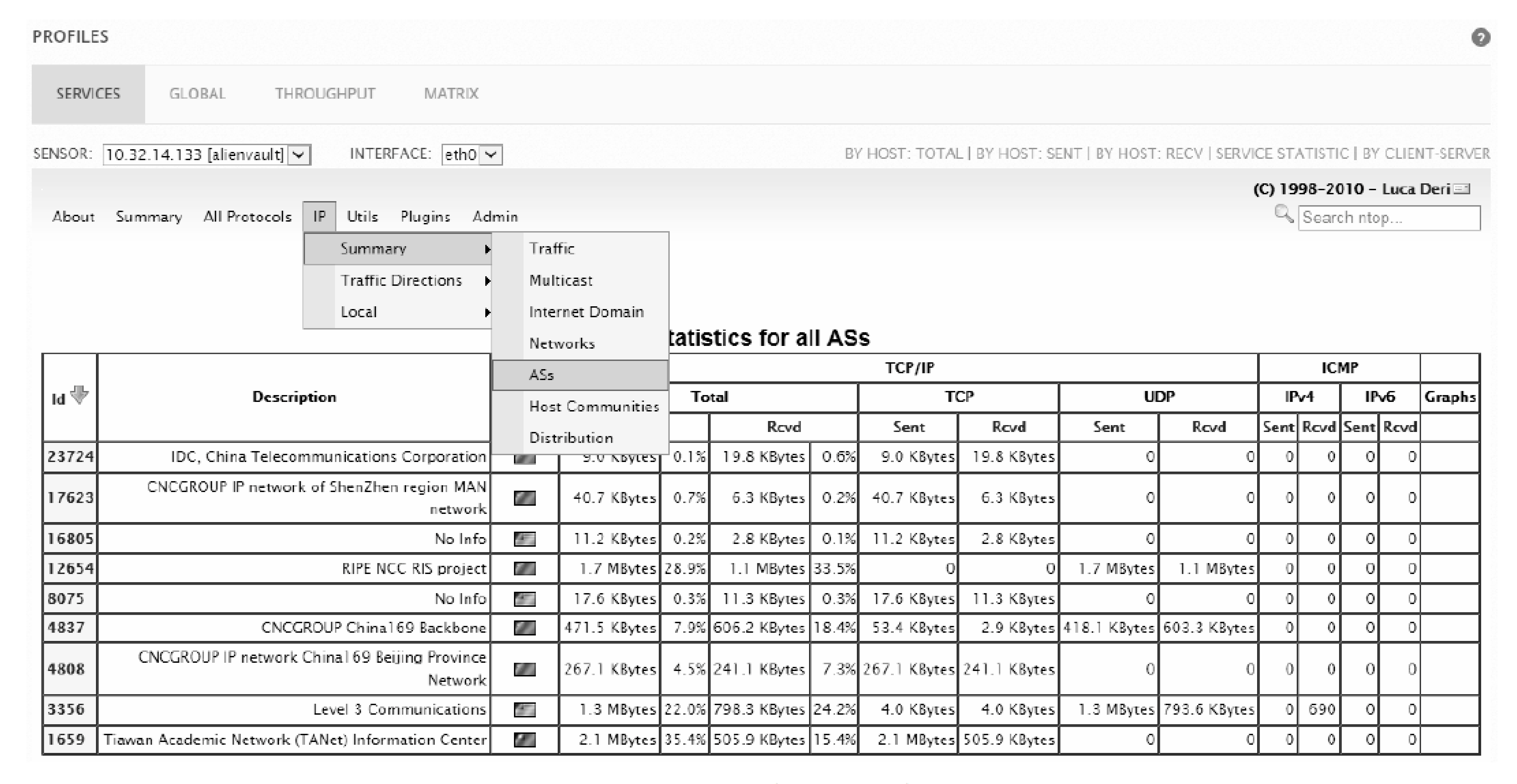


图 8-28　查看自治区的信息

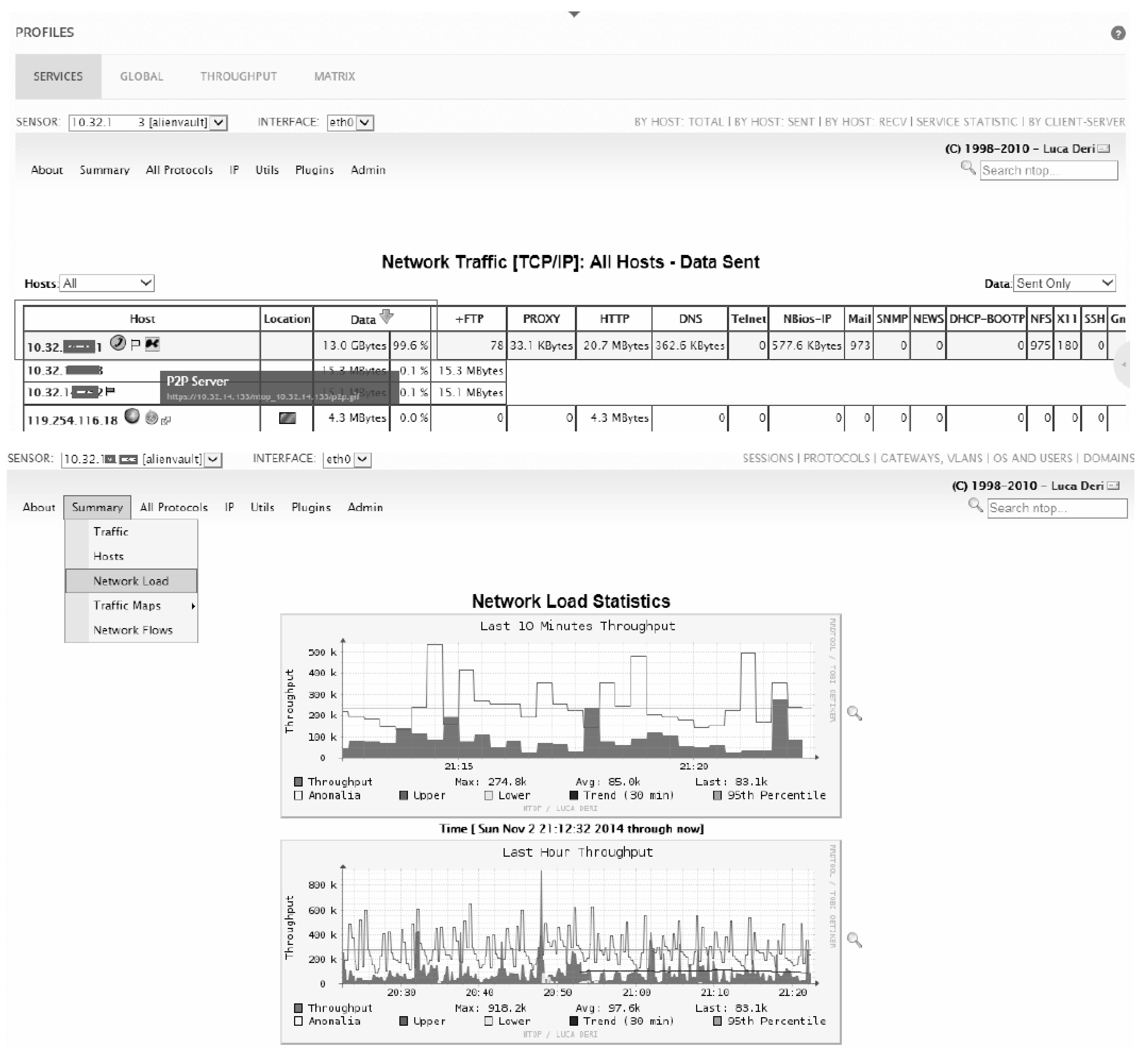


图 8-29　找出流量最高的主机

Active TCP/UDP Sessions

Client	Server	Data Sent	Data Rcvd	Active Since	Last Seen	Duration	Inactive	Latency	Note
192.168.2.251 :54808	pix04.revsci.net :http	968	0	Wed Jan 25 03:03:47 2006	Wed Jan 25 03:04:56 2006	1:09	51 sec		
192.168.2.251 :54750	pixel.ad.mlnadvertising.com :http	783	0	Wed Jan 25 03:02:27 2006	Wed Jan 25 03:03:30 2006	1:03	2:17		
192.168.2.251 :54777	dis.as.criteo.com :http	998	0	Wed Jan 25 03:02:46 2006	Wed Jan 25 03:03:51 2006	1:05	1:56		
192.168.2.251 :54748	www.facebook.com :http	938	0	Wed Jan 25 03:02:27 2006	Wed Jan 25 03:03:30 2006	1:03	2:17		
192.168.2.251 :54794	www.facebook.com :http	951	0	Wed Jan 25 03:03:31 2006	Wed Jan 25 03:04:56 2006	1:25	51 sec		
192.168.2.251 :54839	www.facebook.com :http	1.0 KB	0	Wed Jan 25 03:04:59 2006	Wed Jan 25 03:05:34 2006	35 sec	13 sec		
192.168.2.251 :54751	www.facebook.com :https	1.6 KB	0	Wed Jan 25 03:02:27 2006	Wed Jan 25 03:03:30 2006	1:03	2:17		
192.168.2.251 :54796	www.facebook.com :https	1.4 KB	0	Wed Jan 25 03:03:32 2006	Wed Jan 25 03:04:56 2006	1:24	51 sec		
192.168.2.251 :54834	www.facebook.com :https	3.2 KB	0	Wed Jan 25 03:04:58 2006	Wed Jan 25 03:05:34 2006	36 sec	13 sec		

图 8-30　查看激活端口

TCP/UDP: Local Protocol Usage

Reporting on actual traffic for 8 host(s) on 5 service port(s)

Service		Clients	Servers
domain	53	• pc1-36 [NetBIOS] • 192.168.2.251	
http	80	• 192.168.2.251	
hosts2-ns	81	• 192.168.2.251	
netbios-dgm	138	• pc1-36 [NetBIOS]	• pc1-36 [NetBIOS]
https	443	• pc1-36 [NetBIOS] • 192.168.2.251	

The color of the host link indicates how recently the host was FIRST seen

0 to 5 minutes　5 to 15 minutes　15 to 30 minutes　30 to 60 minutes　60+ minutes

图 8-31　本地协议使用情况

Ntop 中 Data Dump 含义如表 8-8 所示。

表 8-8　Data Dump

表项			含义
Report Type	Hosts		所有主机信息
	Hosts Matrix		导出各个主机之间的流量情况
	Network Interfaces		导出监控接口的信息
	Network Flows		导出已经配置过的 Network 信息
Description			对 Report Type 中各个子项的描述
Action	Format	text	以文本格式导出相应数据
		xml	以 XML 格式导出相应数据
		perl	以 Perl 格式导出相应数据
		php	以 PHP 格式导出相应数据
		python	以 Python 格式导出相应数据
	Attributes List	Long	以长型格式导出数据，一般只对 text 格式有区别
		Short	以短型格式导出数据，一般只对 text 格式有区别
	Dump Data		单击“Dump Data”按钮导出数据

8.4.3　数据大小分析

当前网络中日益盛行的碎片攻击和超长数据包攻击比较常见，防范困难。我们知道在以太网中，数据包的长度在 64～1518 字节之间，多数是几百字节。所以，如果网络中出现过多小于等于 64 字节或大于等于 1518 字节的数据包，表示网络可能遭受攻击，网络管理人员应立即

对网络进行检测分析，以确保网络的安全。

在以太网中，如果数据包小于 64 字节，这称为碎片帧；大于 1518 字节，称为巨人帧。而碎片和巨人帧都是不正常的数据包，它们若大量存在，将影响网络的正常运行，比如过多的碎片将增加网络的负载，过多特大数据包导致网络瘫痪等。为避免网络遭受碎片或特大数据包的攻击，网络管理人员应该对网络中传输的数据包进行检查分类统计，数据包分类如表 8-9 所示。

表 8-9　数据包分类显示

名 称	大小或比例	
Shortest	42 bytes	
Average Size	184 bytes	
Longest	16-114 bytes	
Size <= 64 bytes	59.3%	4,411
64 < Size <= 128 bytes	31.8%	2,367
128 < Size <= 256 bytes	14.5%	1,079
256 < Size <= 512 bytes	2.3%	172
512 < Size <= 1024 bytes	8.5%	630
1024 < Size <= 1518 bytes	4.1%	302
Size > 1518 bytes	5.2%	386

8.4.4　流量分析

NTOP 系统中，对于网络整体流量的统计，分别是 Protocol Traffic Counters、IP Traffic Counters、TCP/UDP Connections Stats、Active TCP Connections List、Peers List。可依不同的 Packet，将流量数据放到不同计算器中。对网络整体流量进行分类统计，包括下列情形：

（1）流量分布：区分为本网络主机之间、本网络与外部网络之间、外部网络与本网络之间的网络流量统计；

（2）数据包分布：依据数据包大小、广播形态和分类及统计；

（3）协议使用及分布：本网络各主机传送与接收数据所使用的通信协议种类与数据传输量。另外，通过 Summary→Traffic 查看整体流量，网络流量会清晰地显示出来。

1. 查看通信协议

数据包形态对于网络安全分析具有至关重要的意义。比如，网络中最常见的数据包既有 TCP 也有 UDP 等类型，如果想了解某台计算机传输了哪些数据，可以双击计算机名称即可分析出用户各种网络传输的协议类型和占用带宽的比例。当发生异常，通过图形直观地展示出来，例如在出现蠕虫攻击时会出现异常流量，其中一种情况如图 8-32 所示。

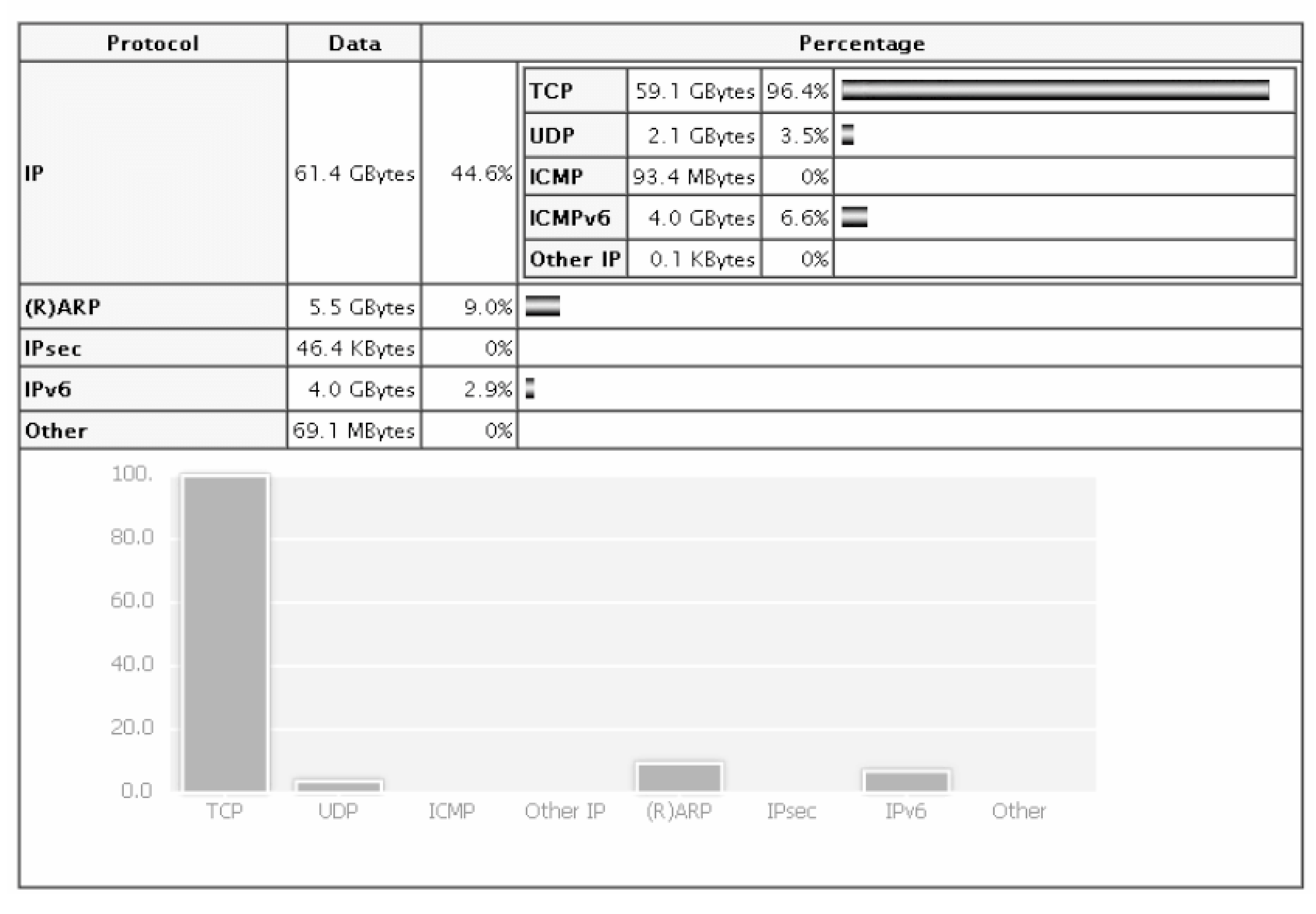
Global Protocol Distribution

Protocol	Data	Percentage				
IP	13.7 GBytes	100.0%	TCP	247.7 MBytes	1.8%	
			UDP	13.4 GBytes	98.2%	
			ICMP	1.4 MBytes	0%	
			ICMPv6	1.1 KBytes	0%	
			Other IP	3.7 KBytes	0%	
IPv6	49.5 KBytes	0%				

100.0
80.0
60.0
40.0
20.0
0.0
TCP
UDP
ICMP
Other IP
IPv6

图 8-32　发生异常流量时协议分布

而正常工作状态下的协议分布主要是 TCP 协议，如图 8-33 所示。



Global Protocol Distribution

Protocol	Data	Percentage				
IP	61.4 GBytes	44.6%	TCP	59.1 GBytes	96.4%	
			UDP	2.1 GBytes	3.5%	
			ICMP	93.4 MBytes	0%	
			ICMPv6	4.0 GBytes	6.6%	
			Other IP	0.1 KBytes	0%	
(R)ARP	5.5 GBytes	9.0%				
IPsec	46.4 KBytes	0%				
IPv6	4.0 GBytes	2.9%				
Other	69.1 MBytes	0%				

图 8-33　正常流量协议分布

2. 查看网络流量图（Local Network Traffic Map）

Ntop 有个很有趣的功能，它能够动态显示网络数据的流量及流向，要实现该功能，首先选择 Ntop 菜单中 Admin→configure→Preference，配置 dot.path 的参数为/usr/bin/dot，如图 8-34 所示。然后在 Ntop 菜单中依次选择 IP→Local→Network Traffic Map。

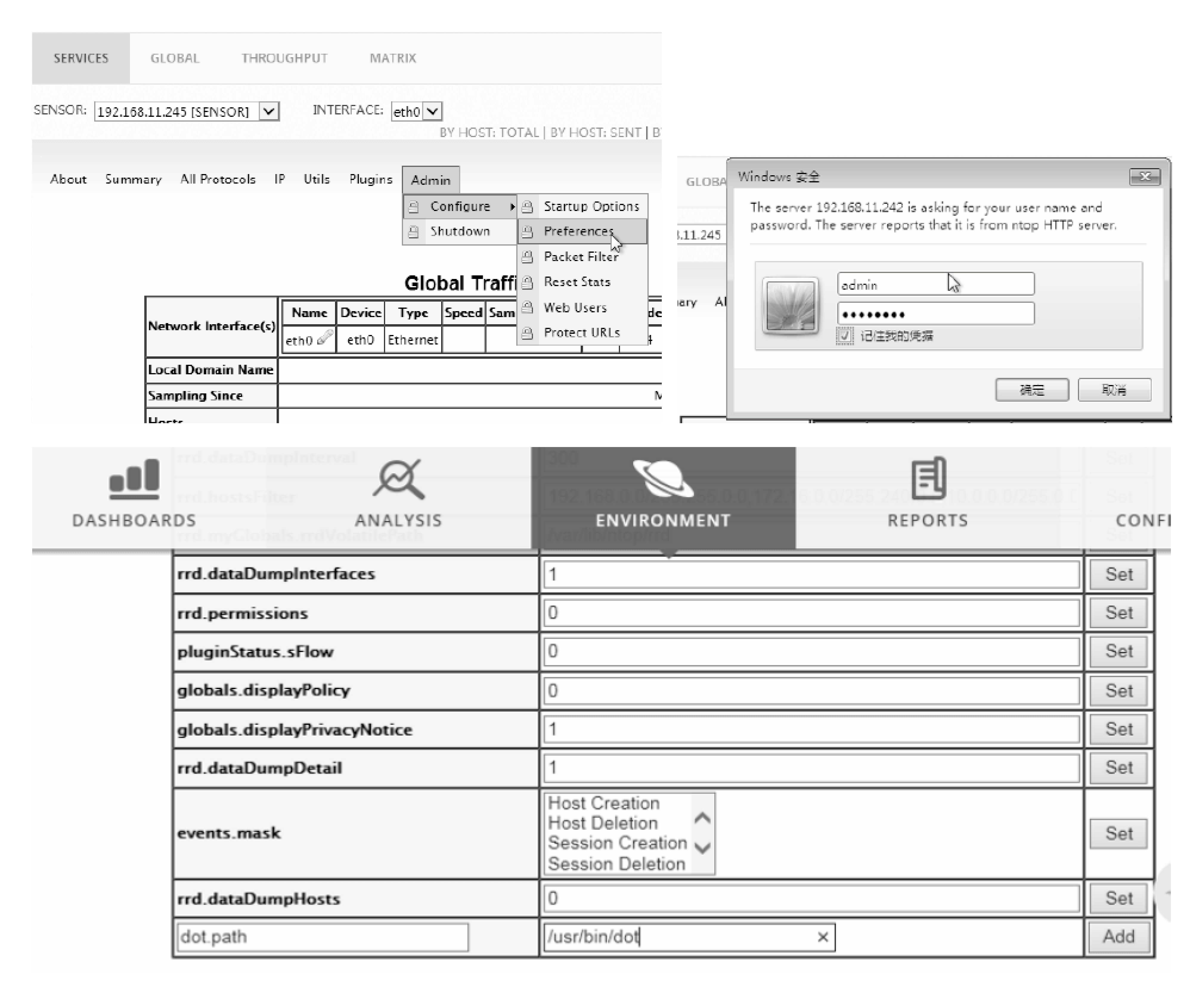


图 8-34 为流量向图设置 dot 路径

这时，可看到一张反映各个主机流量流向的拓扑图，图中箭头方向代表数据的流向，鼠标单击相应 IP 地址还能看到详细的 IP 统计信息。图 8-34 是 Ntop 根据网络流量情况自动生成的拓扑图，此图为系统自动生成反映了数据流向，并会随着流量变化定时更新。

故障举例：有时候在按上述方法设置后，查看 Loca Network Traffic Map 时，没有图像显示，而出现了一行提示"Pleas enable make sure that the ntop html/ directory is properly installed"，同时伴随 404 错误及"Autosuggest is already set!"提示，大家可以尝试以下方法解决：

```
#chown -R ntop:ntop /var/lib/ntop/
#chown -R ntop:ntop /usr/share/ntop/
#ln -s /usr/share/ntop/html  /var/lib/ntop/
#/etc/init.d/ntop restart
```

服务重启完成后需要等待几分钟，不要立刻打开流向图观察。而且在分布式 OSSIM 环境中，具有多个 Sensor 和多个网络接口，在 Ntop 界面显示中和修改配置文件时，一定要区分对应 Sensor 下的网卡，切勿张冠李戴！例如在图 8-34 中，Sensor 的 IP 地址为 192.168.11.245，那么就需要在这台主机下重置 Ntop 的密码，然后进入相应的 Admin 参数配置界面。

图 8-35 反映的是监控网段 192.168.11.0/24 中客户机和 OSSIM（192.168.11.108）交互的数据流向。

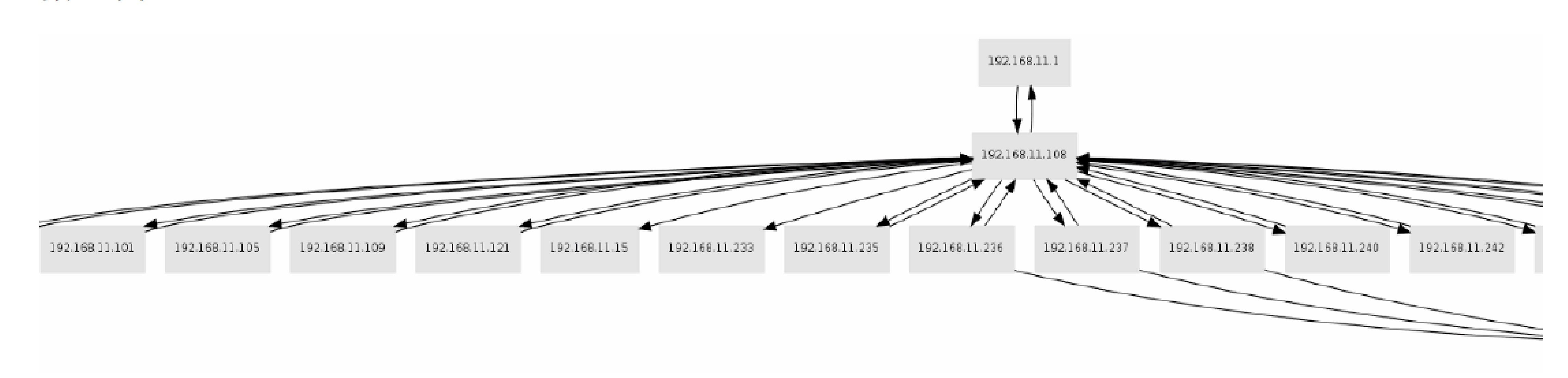


图 8-35　简单网络中的流向图

当网络环境比较复杂时，这种图像会变得很复杂，如图 8-36 所示。

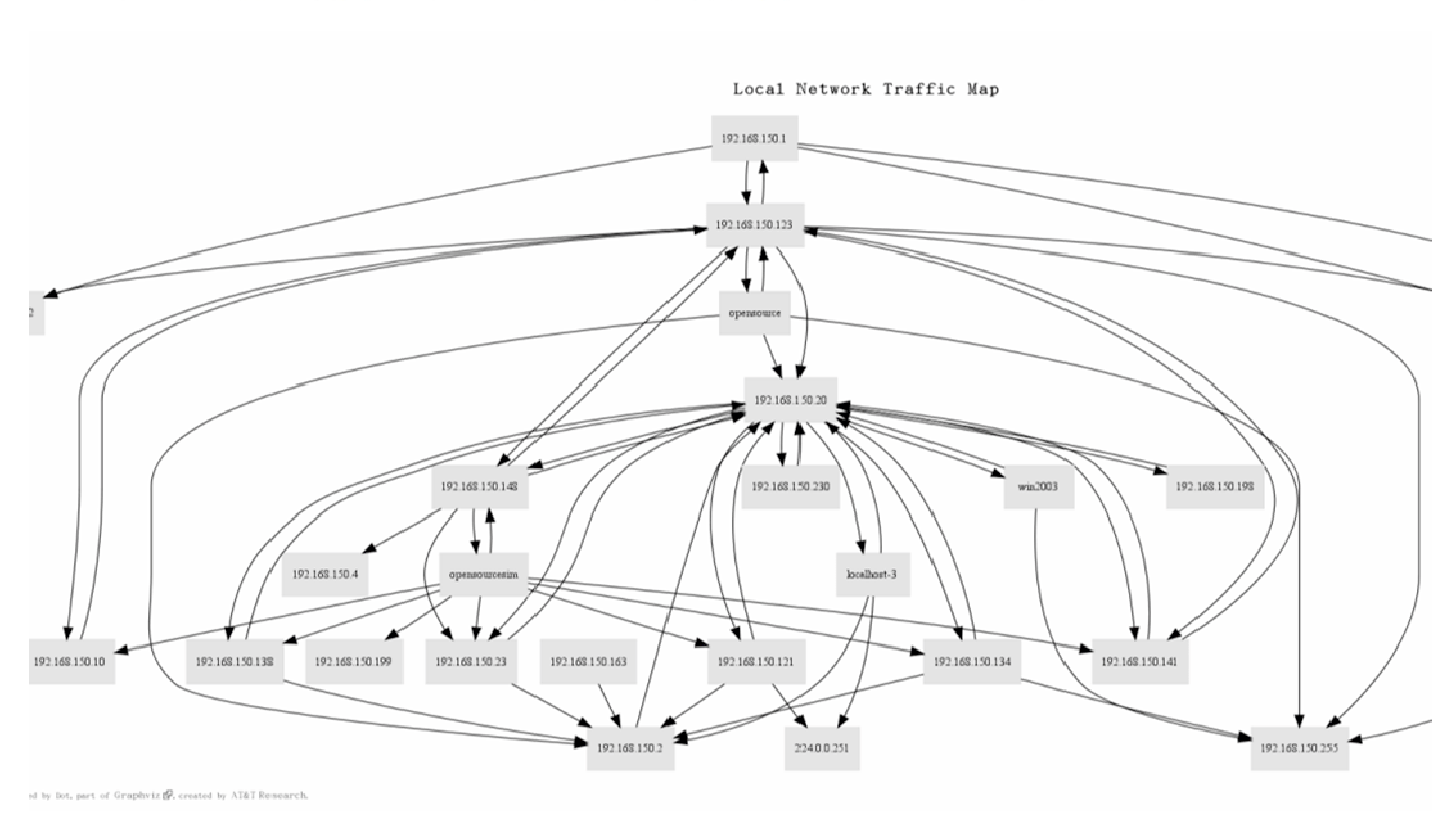
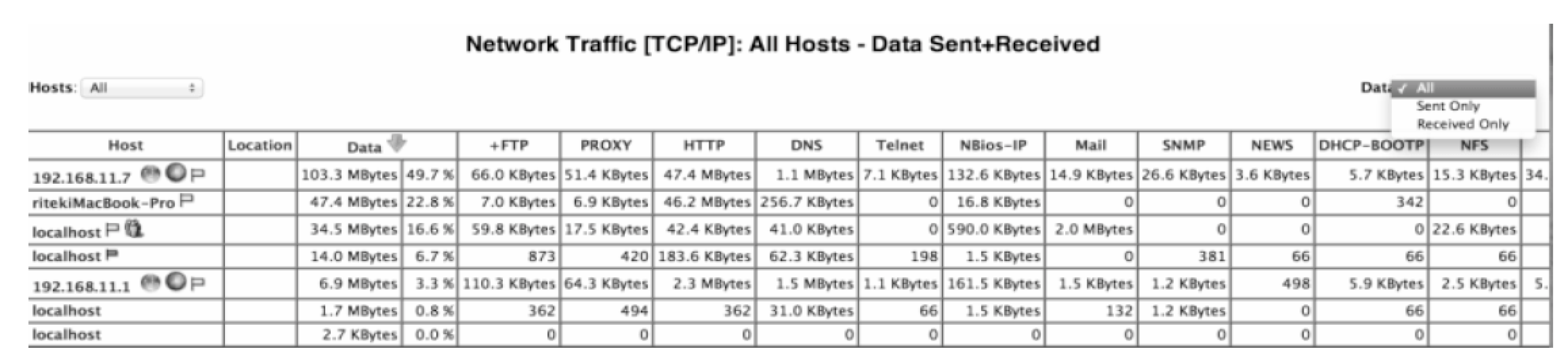


图 8-36　复杂环境中的流向图

Ntop 这张图就是通过开源的可视化分析软件 graphviz 实现，它是通过 digraph 绘制的一种带箭头的有向图。Graphviz 提供了 dot 语言来编写绘图脚本。当设置好 NTop 后，这张图保存在/var/lib/ntop_db_64/目录下，文件名为 network_map.png，系统默认 120 刷新一次，当系统受到 DOS 类攻击时，这张图会密密麻麻地显示着 IP 地址，图片尺寸也会很大。

3. 查看主机流量

在得知网络整体流量信息后，希望深入分析网络中的主机流量情况，从而进行流量限制等方面的管理工作，我们可在 Ntop 中依次选择菜单 IP→Summary→Traffic，出现如图 8-37 所示界面。



Network Traffic [TCP/IP]: All Hosts - Data Sent+Received

Hosts: All

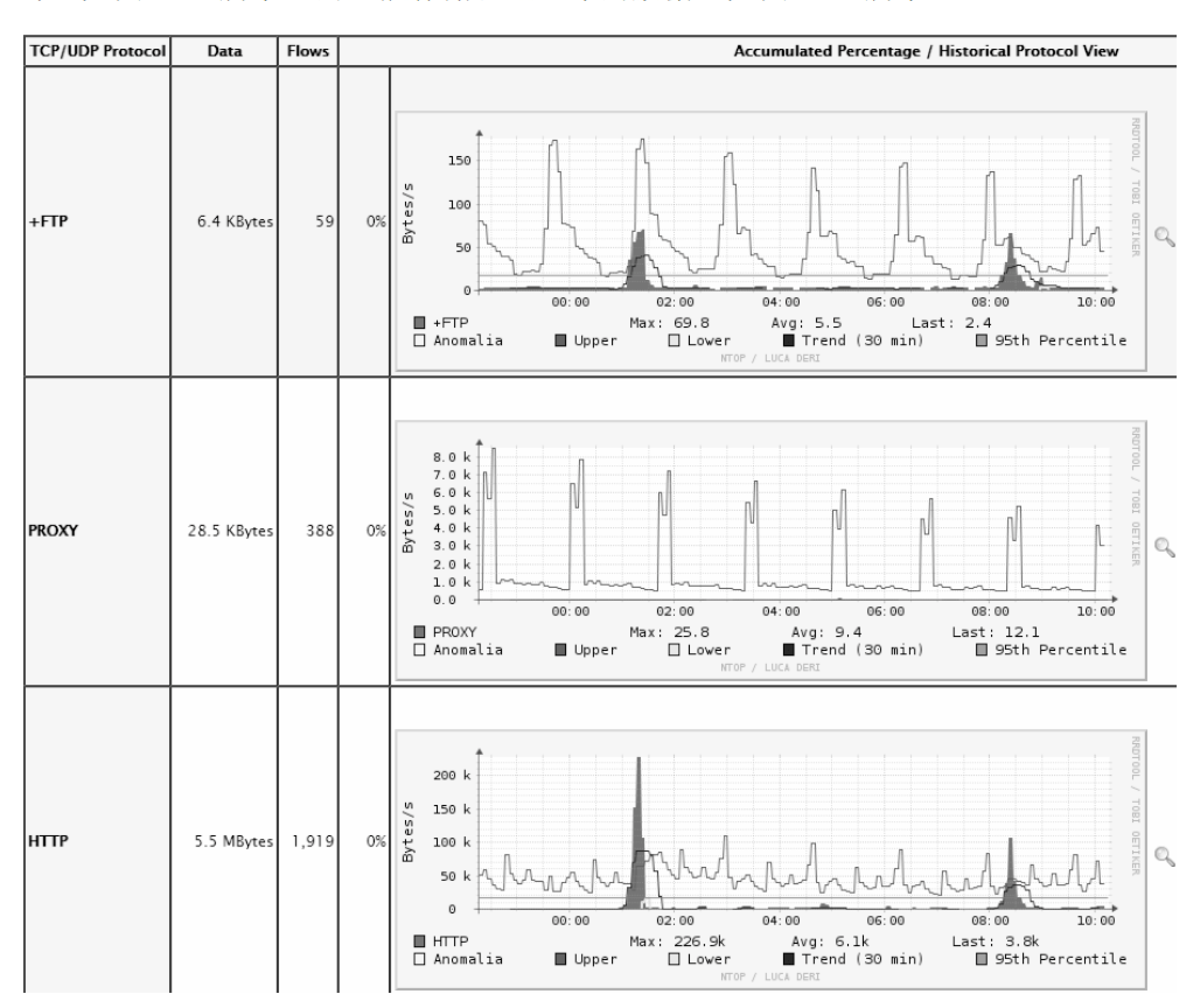
Data: ✓ All / Sent Only / Received Only

Host	Location	Data		+FTP	PROXY	HTTP	DNS	Telnet	NBios-IP	Mail	SNMP	NEWS	DHCP-BOOTP	NFS	
192.168.11.7		103.3 MBytes	49.7 %	66.0 KBytes	51.4 KBytes	47.4 MBytes	1.1 MBytes	7.1 KBytes	132.6 KBytes	14.9 KBytes	26.6 KBytes	3.6 KBytes	5.7 KBytes	15.3 KBytes	34.
ritekiMacBook-Pro		47.4 MBytes	22.8 %	7.0 KBytes	6.9 KBytes	46.2 MBytes	256.7 KBytes	0	16.8 KBytes	0	0	0	342	0	
localhost		34.5 MBytes	16.6 %	59.8 KBytes	17.5 KBytes	42.4 KBytes	41.0 KBytes	0	590.0 KBytes	2.0 MBytes	0	0	0	22.6 KBytes	
localhost		14.0 MBytes	6.7 %	873	420	183.6 KBytes	62.3 KBytes	198	1.5 KBytes	0	381	66	66	66	
192.168.11.1		6.9 MBytes	3.3 %	110.3 KBytes	64.3 KBytes	2.3 MBytes	1.5 MBytes	1.1 KBytes	161.5 KBytes	1.5 KBytes	1.2 KBytes	498	5.9 KBytes	2.5 KBytes	5.
localhost		1.7 MBytes	0.8 %	362	494	362	31.0 KBytes	66	1.5 KBytes	132	1.2 KBytes	0	66	66	
localhost		2.7 KBytes	0.0 %	0	0	0	0	0	0	0	0	0	0	0	

图 8-37　根据协议分类的主机流量

8.4.5　协议分析

Ntop 基于嗅探模式工作除了能识别 2~3 层信息，还能识别应用层协议，如 Web、Ftp、Samba、IRC 等，如图 8-38 所示，而且能存储长达一年的数据，如图 8-39 所示。



TCP/UDP Protocol	Data	Flows		Accumulated Percentage / Historical Protocol View
+FTP	6.4 KBytes	59	0%	
PROXY	28.5 KBytes	388	0%	
HTTP	5.5 MBytes	1,919	0%	

图 8-38　识别应用层流量

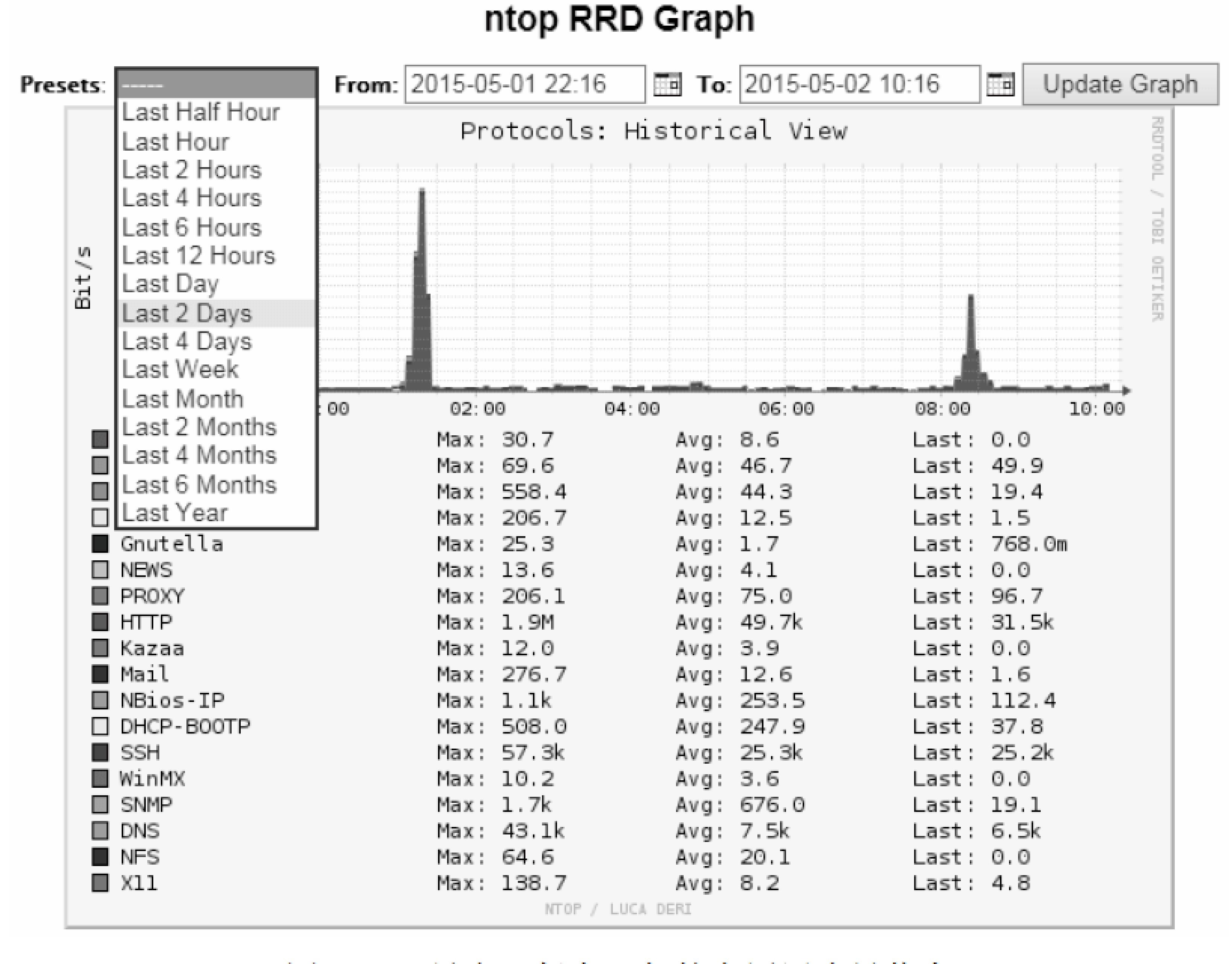


图 8-39　最多可保存一年的应用层流量信息

8.4.6　负载分析

在分布式 OSSIM 系统中，通过查看不同的 Sensor，并选择 Summary→Network Load 菜单查看监控网段的负载情况，包括历史负载数据的查询，都能通过单击 🔍 图标选择相应时间段即可实现。如图 8-40 所示。

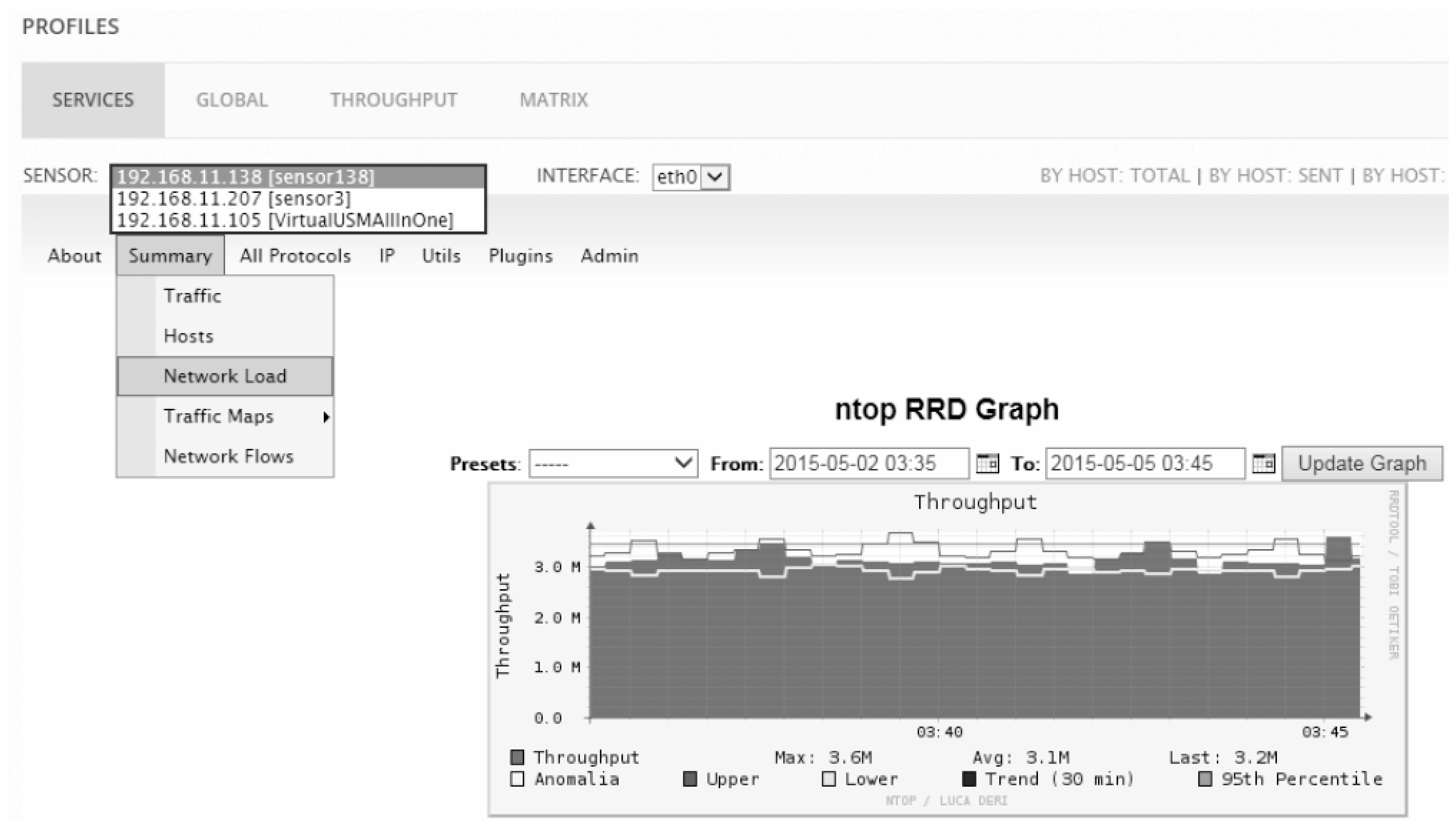


图 8-40　不同 Sensor 下的负载分析

8.4.7 Ntop 应用于网络视频监视

近年来使用迅雷看看、PPS 等在线视频点播已经成为消耗企业网络带宽的主要应用。由于这些主机上装有一些百度、迅雷和暴风影音的插件，在内网连接的模式下，这些插件自动与外网的地址进行连接，在域名查询的时候，将请求发送到了无法解析外网地址的内网 DNS，因此返回了查询错误。这些大量的 DNS 请求查询和返回错误的数据包不但经过公司内部的交换机，还经过路由器和 Internet 出口链路而占用宝贵的网络资源，对网络通信质量产生了较大的影响，在 Ntop 上的监控图像，如图 8-41 所示。

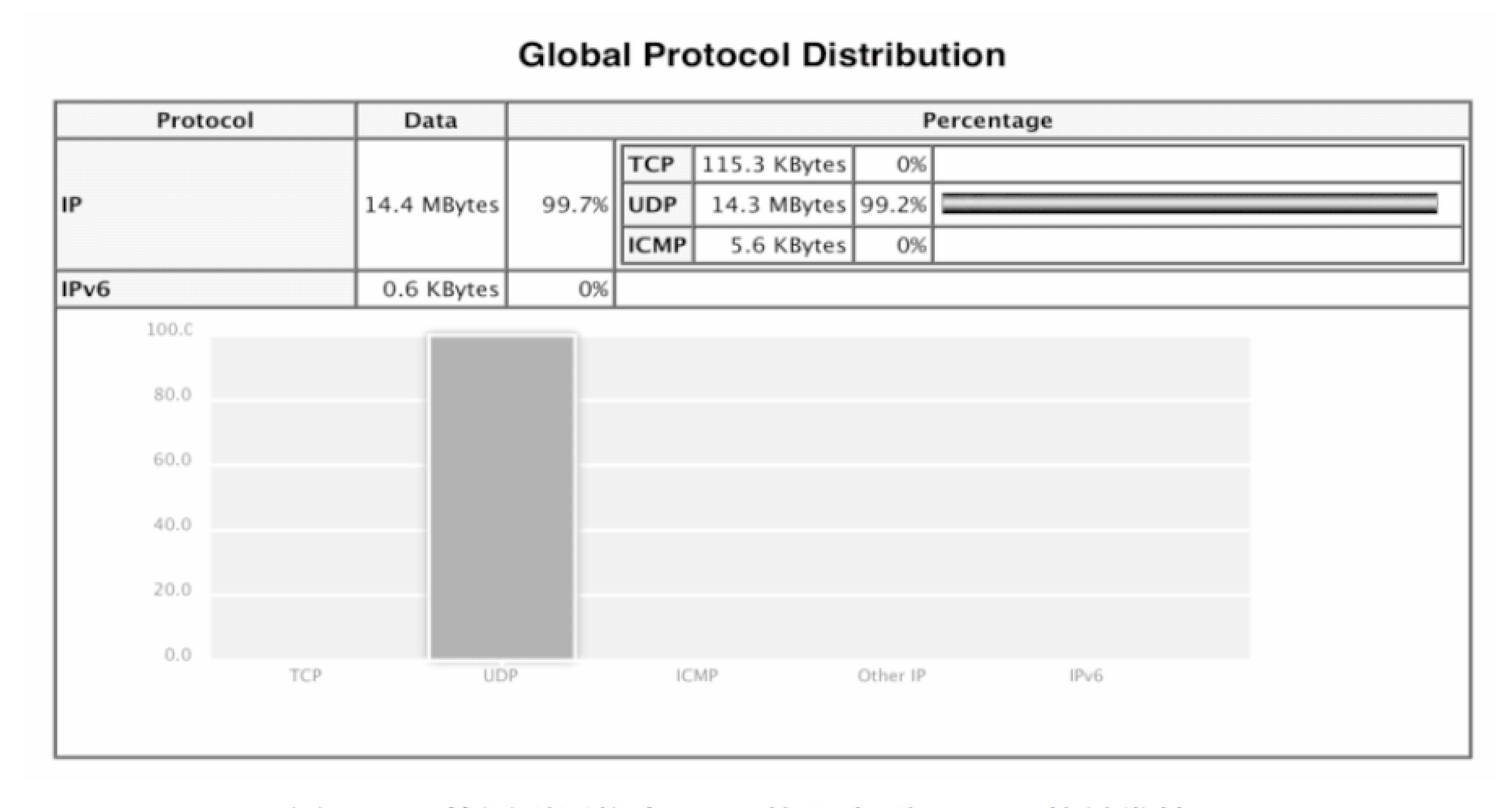
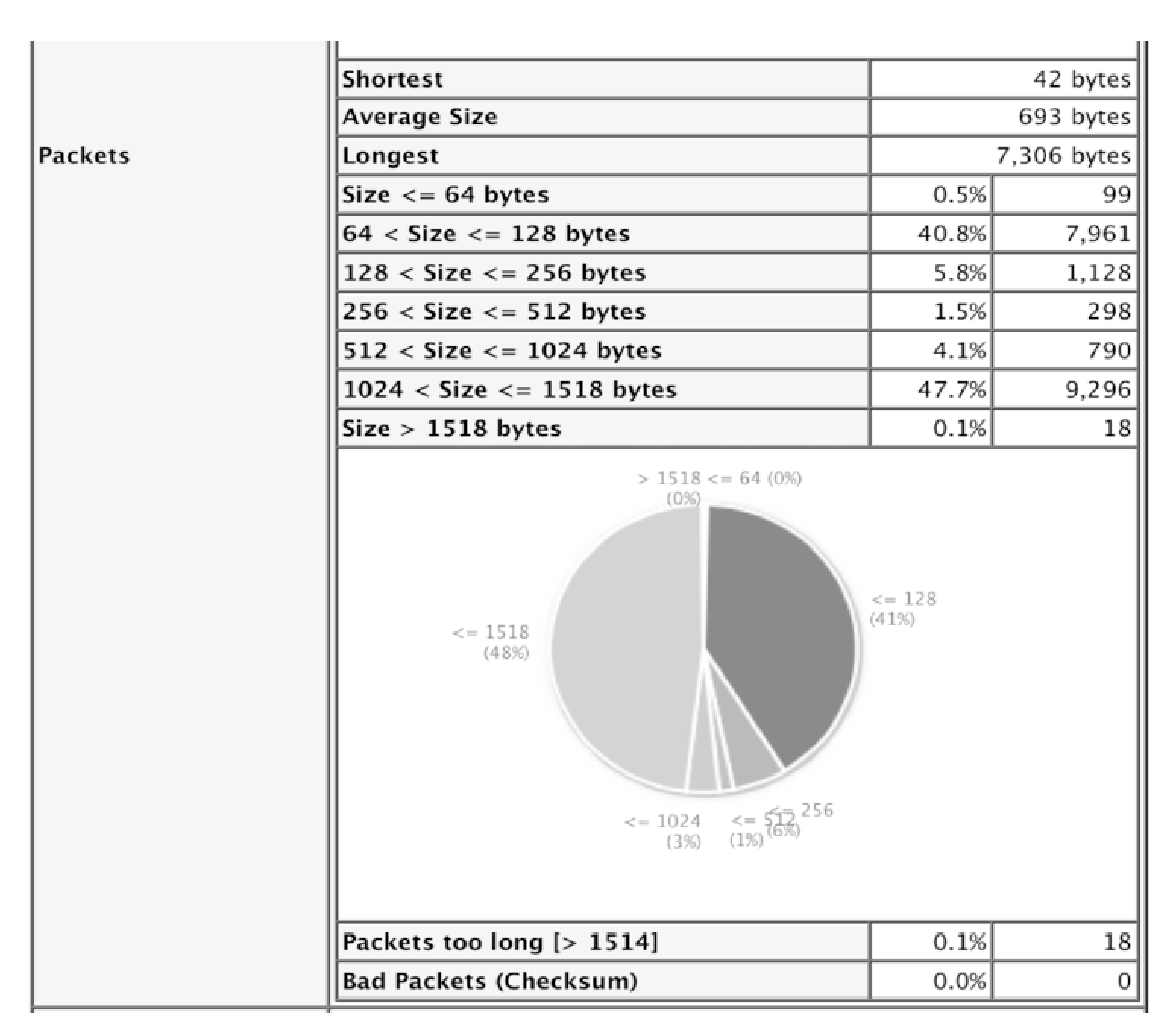


图 8-41　检测到网络中 UDP 协议占到 99.2% 的异常情况

下面我们通过 OSSIM 中集成的 Ntop 工具，研究迅雷看看的数据特征。首先，在经过一段时间的抓包分析后，我们打开 Ntop 界面，发现默认情况下主机使用 UDP 的 12128 端口在与外部大量的 IP 进行数据传输，而且网络中 UDP 的数据包占绝大部分。

我们发现网络利用率在 80%以上，而当关闭视频电影以后，利用率迅速回落到 10%以下。我们可以看到，网络数据包总呈现大小包分布两极化趋势，65~127 数据包占 50%。而 1024~1517 的数据包占了 36%，其余大小的数据包不到 20%。迅雷看看对网络带宽的消耗很大，可以在很短时间内将网络带宽充分占用，这和迅雷看看采用 UDP 通信方式有一定关系，UDP 通信本身没有速度限制，尽最大可能地利用网络带宽的特性，通常出现在线点播这种服务中，如图 8-42 所示。

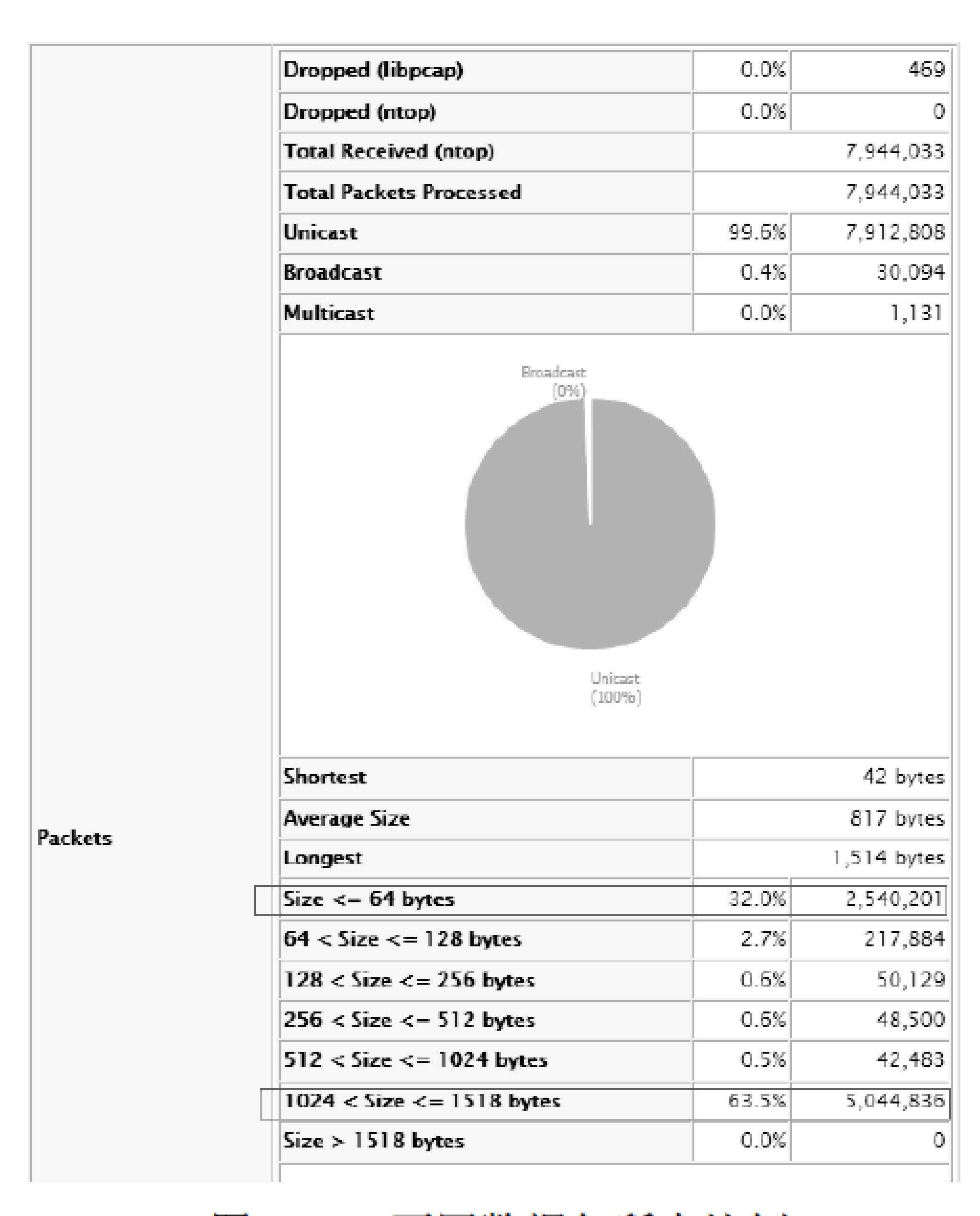


Packets	Shortest		42 bytes
	Average Size		693 bytes
	Longest		7,306 bytes
	Size <= 64 bytes	0.5%	99
	64 < Size <= 128 bytes	40.8%	7,961
	128 < Size <= 256 bytes	5.8%	1,128
	256 < Size <= 512 bytes	1.5%	298
	512 < Size <= 1024 bytes	4.1%	790
	1024 < Size <= 1518 bytes	47.7%	9,296
	Size > 1518 bytes	0.1%	18
	Packets too long [> 1514]	0.1%	18
	Bad Packets (Checksum)	0.0%	0

图 8-42　数据包分布

例子：Ntop 应用-分析数据包大小

首先进入 Environment→Profiles→Global 菜单中，可以得到全局菜单统计信息。而在 eth0 的流量报告中显示不同大小数据包所占比例情况，从而可以找出疑似故障源，如图 8-43 所示。



Packets	Dropped (libpcap)	0.0%	469
	Dropped (ntop)	0.0%	0
	Total Received (ntop)		7,944,033
	Total Packets Processed		7,944,033
	Unicast	99.6%	7,912,808
	Broadcast	0.4%	30,094
	Multicast	0.0%	1,131
	Shortest		42 bytes
	Average Size		817 bytes
	Longest		1,514 bytes
	Size <= 64 bytes	32.0%	2,540,201
	64 < Size <= 128 bytes	2.7%	217,884
	128 < Size <= 256 bytes	0.6%	50,129
	256 < Size <= 512 bytes	0.6%	48,500
	512 < Size <= 1024 bytes	0.5%	42,483
	1024 < Size <= 1518 bytes	63.5%	5,044,836
	Size > 1518 bytes	0.0%	0

图 8-43　不同数据包所占比例

依然是这个界面，在图 8-44 中反映出了监控网络中的协议分布情况。

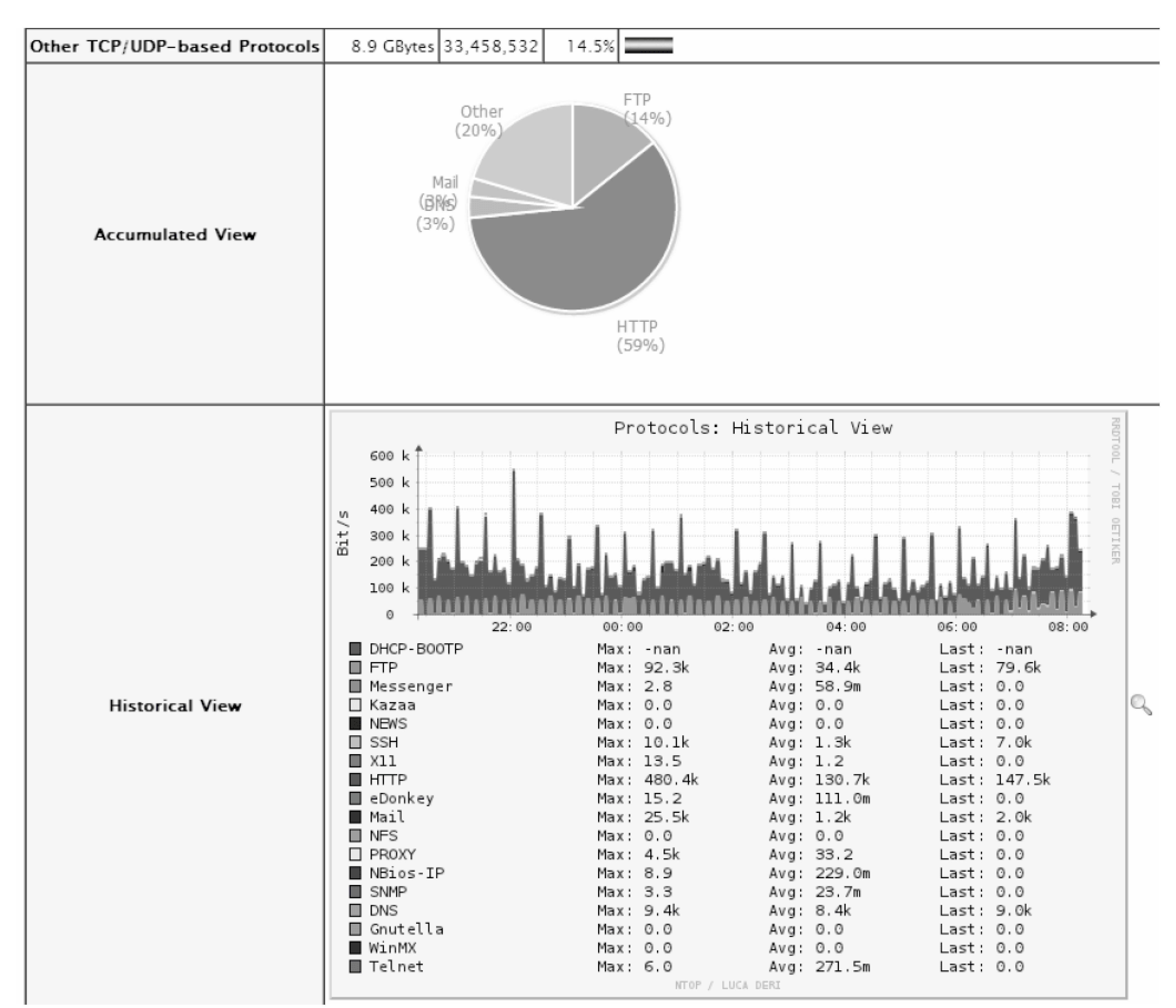


图 8-44　协议分布情况

8.4.8　Ntop 的风险旗帜标示

在本地主机角色图中我们发现，在 Ntop 里用彩旗的颜色来描述风险的高低，在系统里用绿旗、黄旗、红旗分别表示低、中、高风险程度。如图 8-24 和图 8-25 所示，例如有的主机私自修改了 MAC，有的主机发送欺骗 MAC 攻击的数据包，类似这样的主机都会被标记红色旗帜代表高风险。

我们可以使用 Ntop 分析各主机的网络使用量，了解各个主机使用带宽的比例。在 Profiles 中打开 Ntop 界面，选择“Summary/Host”，选择一台主机，例如 alienvault.alienvault，单击后，显示详细流量情况如图 8-45 所示。

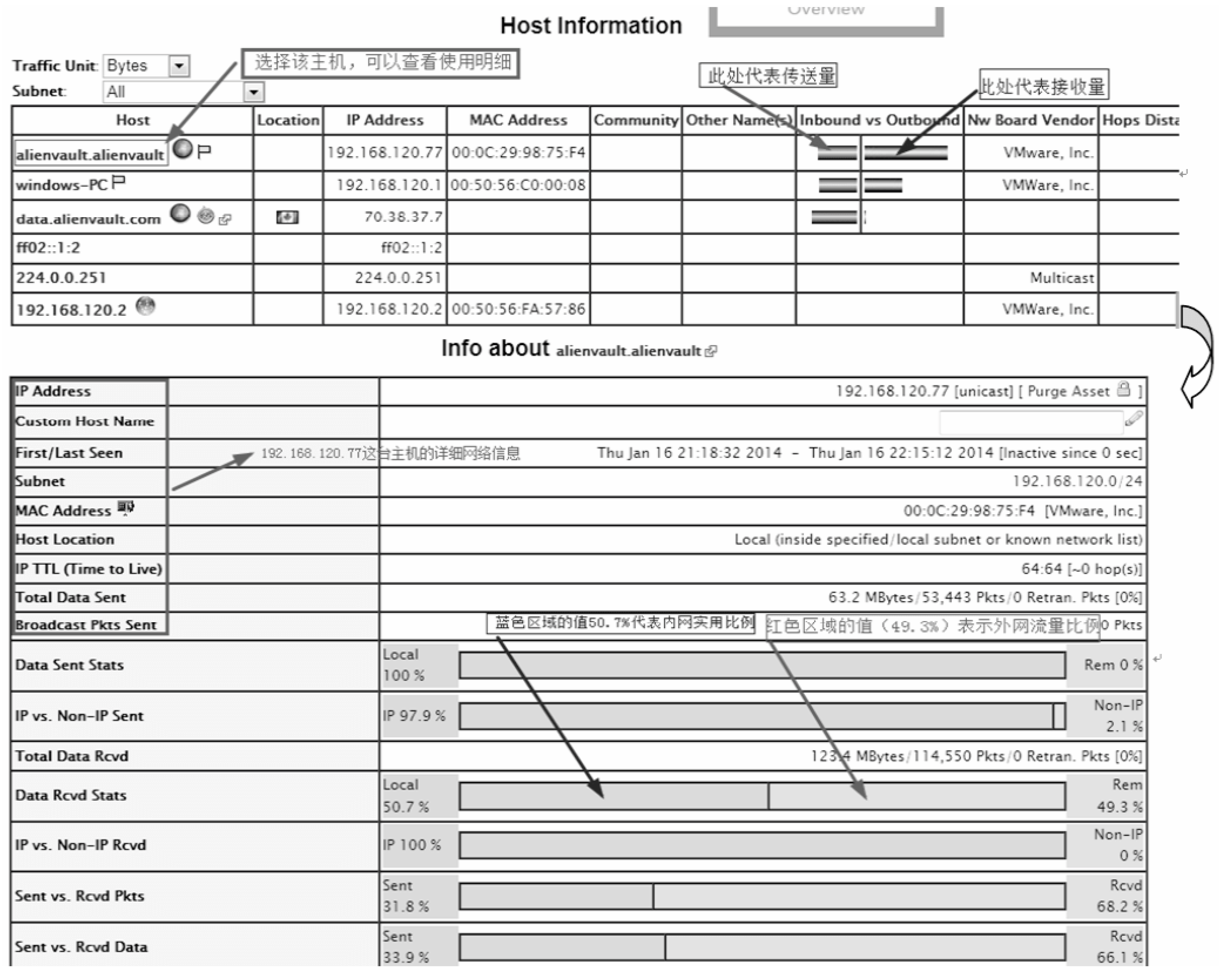


图 8-45　查看主机详情

图 8-45 中，224.0.0.251 代表所有的支持组播的 DNS 服务器，有时候会遇到一个比较特别的多播地址 239.255.255.250，往往伴随占用大量带宽。这是因为有 UPnP 服务存在，而这个服务会用 SSDP（简单服务发现协议）就是用 239.255.255.250 的多播地址端口 1900 来发现 UPnp 服务，解决方法是在路由器或交换机上禁用 UPnP 即可。如果确定网络中没有流媒体服务器就可以大胆地将其禁用。当 SSH 服务器遭受暴力破解攻击时，Ntop 的取样图中数据包大小分布如图 8-46 所示。

在 OSSIM 的仪表盘上能立刻显示出 Recon 报警数量，也就是在图 8-47 中显示饼图面积最大的区域，单击之后会显示出所有 Snort 报警信息。

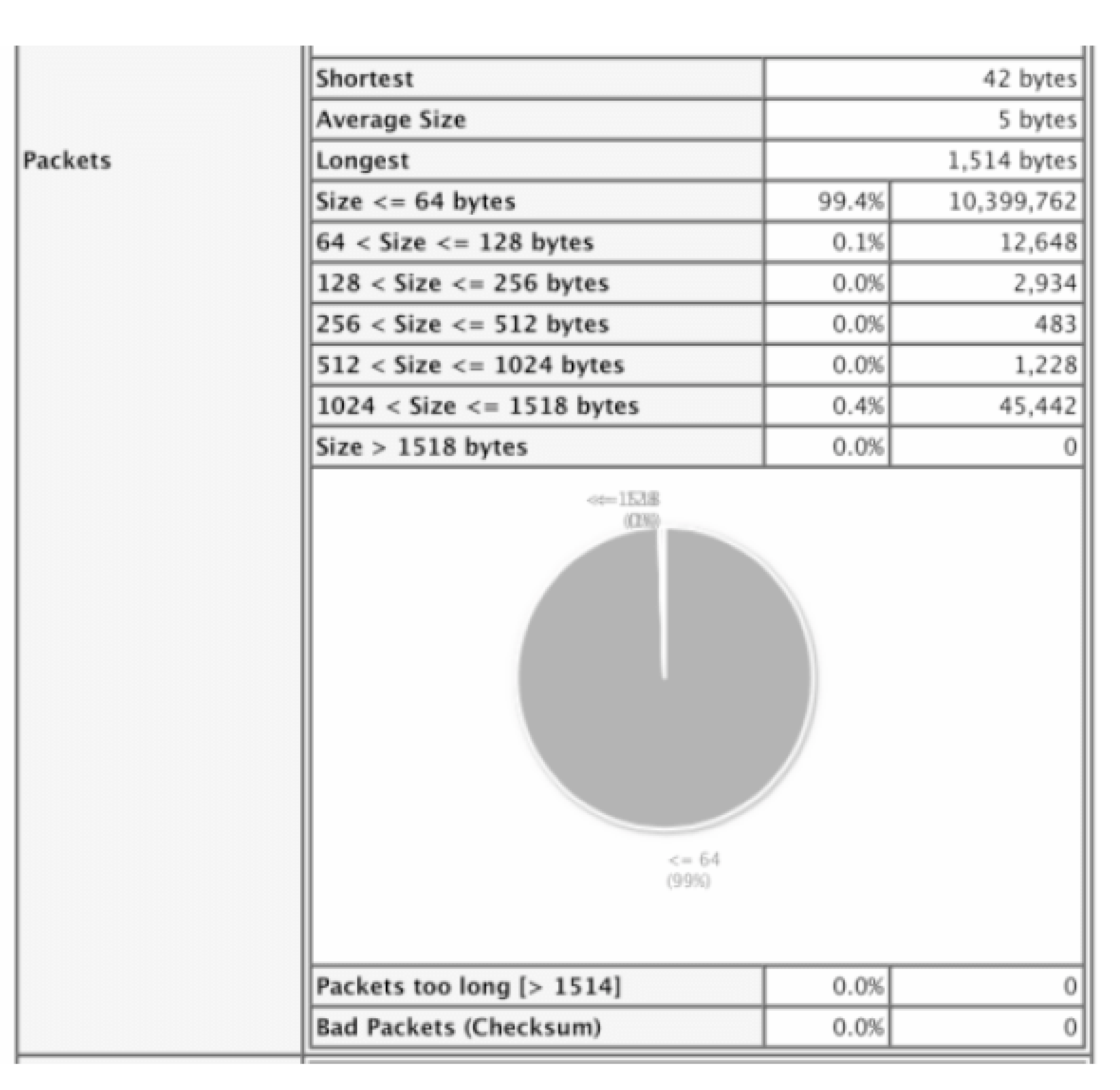


图 8-46　存在大量小于 64 字节的畸形帧

图 8-47　OSSIM 显示的侦测报警

接着，我们单击其中一条 “ET SCAN protential SSH Scan” 事件，查看详情如图 8-48 所示。

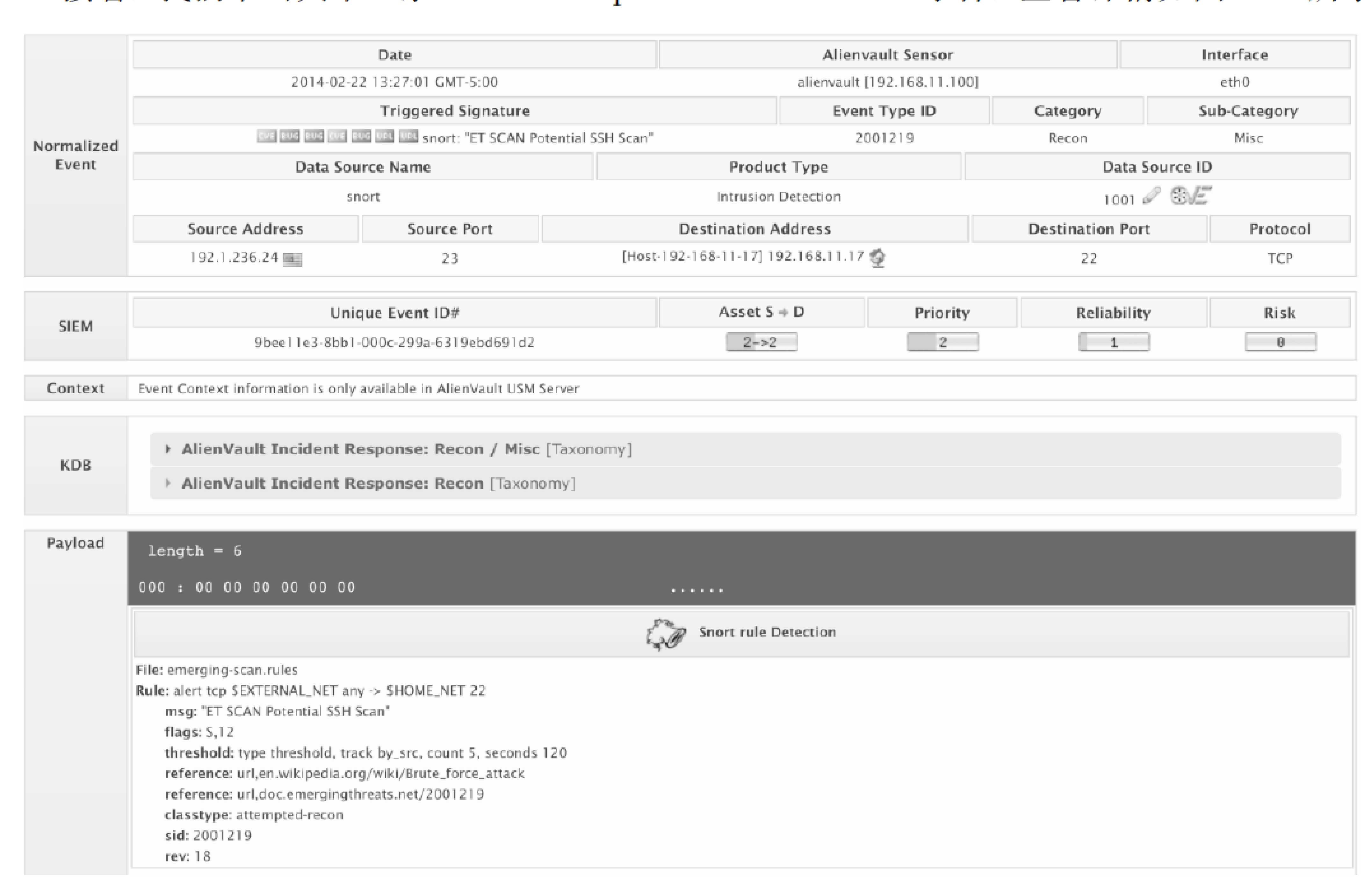


图 8-48　查看其中一条事件的详情

从图中 Payload 看出，这种连接负载（Payload）长度为 6 字节，这不符合常理。Ntop 下

发现蠕虫攻击异常流量，如图 8-49 所示。

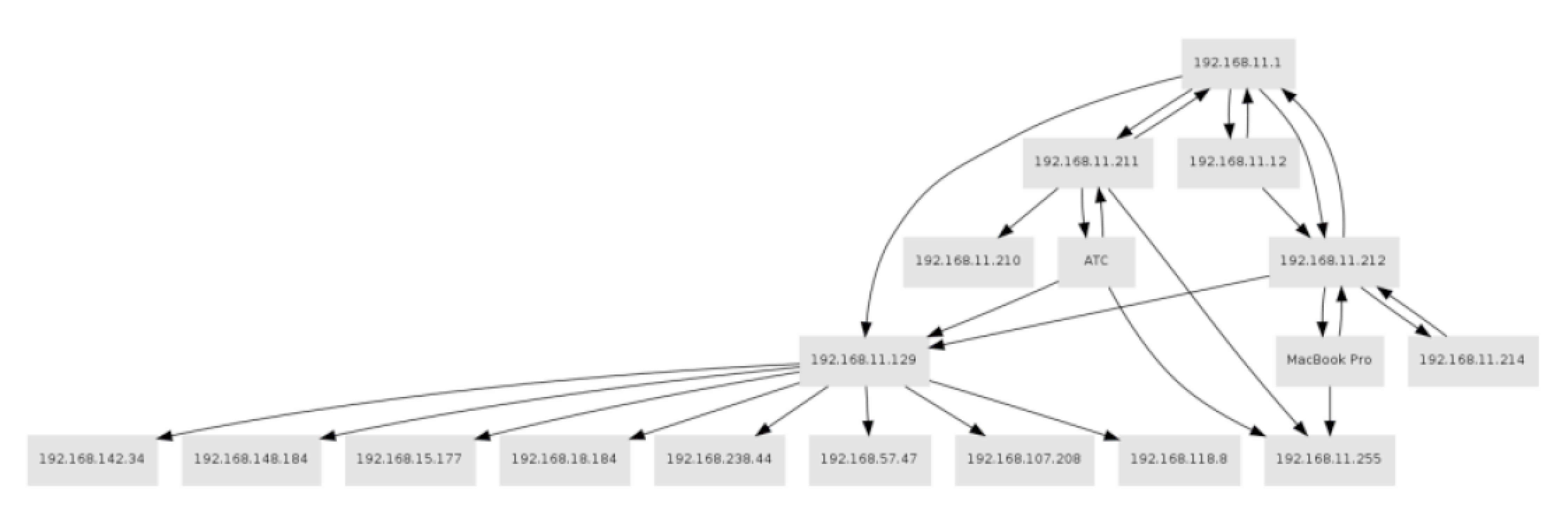


图 8-49　主机（192.168.11.129）正在发送扫描

8.4.9　升级到 Ntopng

Ntopng 是 Ntop 下一代版本，如果你喜欢 Ntopng，则需要从源码包开始编译，操作如下：

（1）准备环境（以 Alienvault USM 4.10 为例）。

```
#apt-get update
#apt-get install gcc
#apt-get install build-essential
#apt-get install libglib2.0 libxml2-dev libpcap-dev libtool autoconf automake
autogen libsqlite3-dev libgeoip-dev subversion subversion-tools
```

（2）下载最新 ntopng 包 http://sourceforge.net/projects/ntop/files/ntopng/，接着解压缩包。

```
#tar -xzvf ntopng-1.2.0_r8116.tar.gz
```

将其移到/opt 目录下：

```
#mv ntopng-1.2.0  /opt/ntopng
#cd /opt/ntopng
```

（3）开始编译。

```
#./autogen.sh
```

执行完第一步之后出现 Now run ./configure 提示，这时在当前目录产生 configure 文件。

```
#./configure
```

执行该命令后显示“You are now ready to compile typing make”说明可以开始编译操作。

```
#make
#make install
```

完成之后会下载 IP 库。

（4）启动 Ntopng。

默认会占用 3000 端口，为了不冲突使用 3001 端口。

```
#./ntopng  -w 3001
```

最后在 OSSIM 防火墙上允许访问 3001 端口，然后就可以打开浏览器输入“服务器 IP:3001”来访问，因为本实验在 OSSIM USM 中完成，所以服务器 IP 即为 OSSIM USM 的 IP 地址。首次登录用户，密码均为 admin Ntop，工作截图如图 8-50 所示。

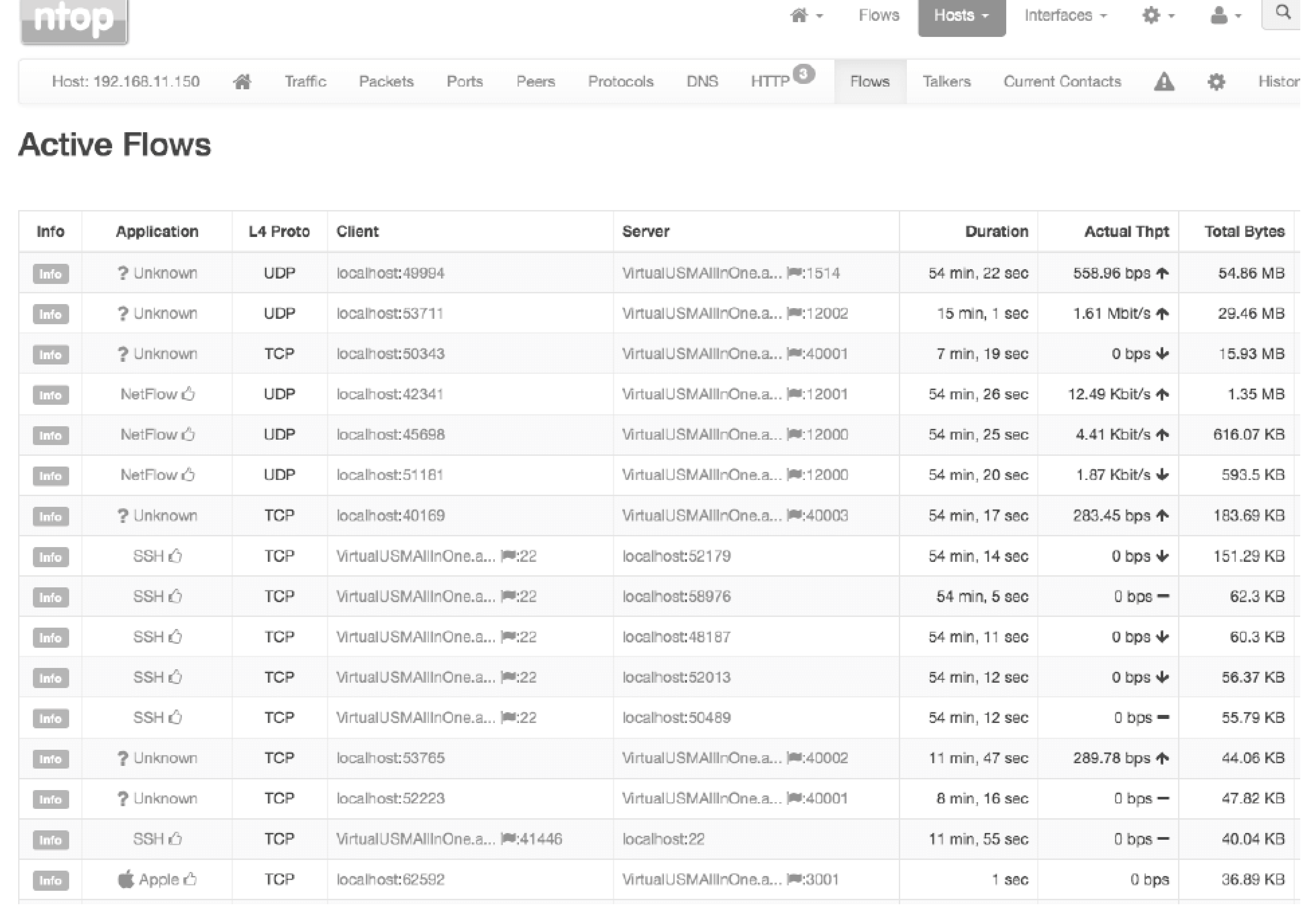


图 8-50　Ntopng 工作界面

8.5 故障排除

8.5.1　多网卡问题

OSSIM 系统中 Ntop 稳定运行的前提是 Sensor 工作状态正常，否则会出现 Ntop 无法探测数据的情况。另外，如果系统中安装了多块网卡，一定要将暂时不使用的网卡去掉，否则也会对 Ntop 运行造成影响，例如出现下列报错就是多网卡造成：“Sensor not available please select for the above dropdown”。一般处理这种情况的方法是，到 Sensor configuration 中，将不用的网卡去掉，即可解决。

网络中传输数据包的大小，将直接反映网络的通信状况，并影响着网络通信质量，所以大家应经常对网络中传输的数据包进行检查，多深入进行分析比对，例如将正常情况下的图表打印出来以便作为参考依据，当出现异常情况时可以迅速进行比对，以避免出现该类故障。

8.5.2 Ntop Web 页面打开缓慢对策

由于在 OSSIM 系统中的 Ntop 模块启动会进行主机域名解析、会话跟踪，当流量较大时，通常 CPU 负载较高，此时页面载入迟缓，我们可以修改/var/lib/ntop/init.cfg 配置文件，方法如下：

```
#vi /var/lib/ntop/init.cfg
```

增加参数--disable-mutexextrainfo，见下面第三行内容：

```
USER="ntop"
INTERFACES="eth0"
GETOPT="-t 1 --no-mac -m \"10.32.14.0/24\" -g -n -z --disable-mutexextrainfo"
```

保存，退出后，再重启 ntop 进程/etc/init.d/ntop restart。

```
#/etc/init.d/ntop restart
```

这里假设您监控的网段为“10.32.14.0/24”。而且每次升级 OSSIM 系统之后都需要检查 init.cfg，如果被修改，就需要改成上文的格式。

参数注释：

- -n：只显示 IP，不解析为域名。
- -z：关闭 TCP 会话追踪，以便获得更好的性能。
- --disable-mutexextrainfo：关闭信号锁定，降低资源消耗。
- --no-mac：不显示网卡 Mac 地址。

8.5.3 “Sensor not available”故障对策

在使用 OSSIM 系统的 Ntop 模块时，出现“Sensor not available，please select from the above drop down”，如图 8-51 所示。

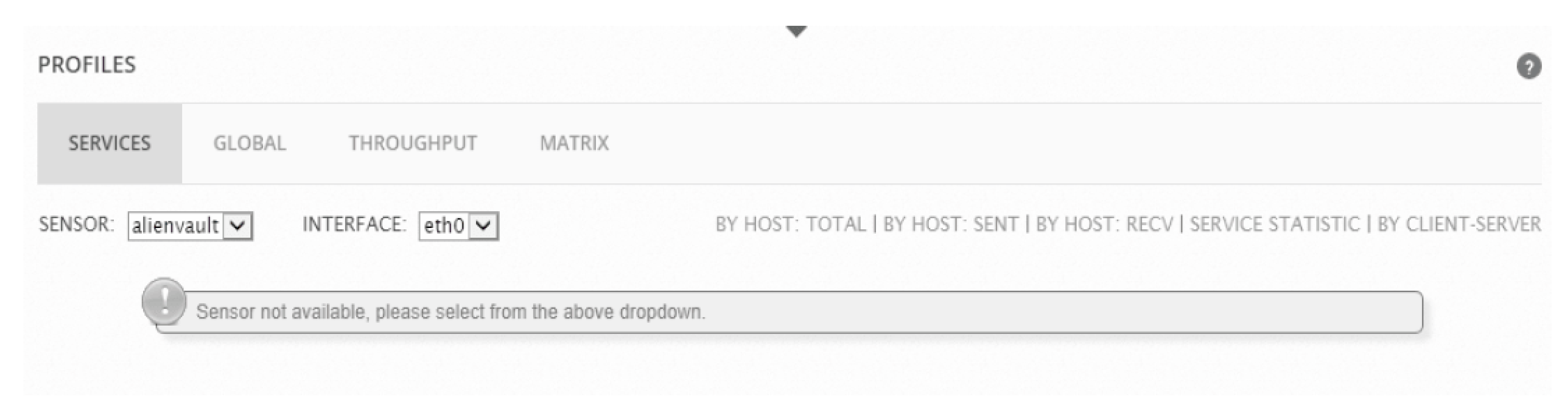


图 8-51　Sensor 失效

当出现这种情况时，单击“GLOBAL（全局）”或“Matrix”按钮，再刷新 Ntop 页面，几秒钟后即可恢复正常。

8.5.4　暂停 Ntop 服务

如果你希望在 Sensor 上停止 Ntop 服务，在 Web UI 界面上没有停止 Ntop 的按钮，只能在命令行下停止。

```
#/etc/init.d/ntop stop
```

其实这样操作并不管用，应为系统启用了 monit 服务，其中 Ntop 的 monit 服务会自动检查 Ntop 是否停止，如果停止会立即重启。在 OSSIM 系统中除了 Ntop 服务之外，还有 alienvault-api、avdatabase、avopenvas、memcache、avagent、avframework、avsensor、nfcapd、avapache、avnagios3、avserver 服务都有这种自动监控功能。这时，临时做法是修改 ntop.monitrc 文件。

```
vi /etc/monit/alienvault/ntop.monitrc
```

停止 ntop 服务即可，但如果执行了 alienvault-update 升级指令，这些服务配置文件又会重置到初始状态。在 OSSIM 5.1.0 之后的发行版中便彻底移除了 NTOP 服务。

8.5.5　管理员密码遗忘对策

很多朋友遇到了不少问题，其中 Ntop 的用户密码文件是经过加密存储在 ntop_pw.db 文件中，Ntop 用户密码存储位置：

- 64 位版本：/var/lib/ntop_db_64/ntop_pw.db
- 32 位版本：/var/lib/ntop/ntop_pw.db

如果需要修改 admin 密码，则分两种情况，32 位版则直接执行 ntop -A ，即能修改密码，如果是 64 位版本，则需先删除其密码文件 ntop_pw.db，然后用 notp -A 重置管理员密码后，最后重启 Ntop 服务（service ntop restart）就能生效。

8.6 用 Nagios 监视

从事运维的工作人员常管理着成百上千台设备，对于流量监控总找不到一款好用的工具，但平日里工作事情很多，总不能天天盯着设备查看流量吧，因此大家最关心的是服务器的监控与报警问题。希望如果机器出故障了，能第一时间发现并处理。发现机器有没有问题，对我们而言不是什么难事。编写脚本 Ping IP 地址以这种方式来检测主机可用性，Telnet/SSH 每台机器的管理端口如果增加了新机器，就修改配置即可。这种原始方式让你管理十几台机器还可以，如果成百上千台服务器呢，不但不好维护和管理，而且也无法打出报表。现实中，很多人对监控程度把握得不好，不假思索地上了一堆的监控项目，往往适得其反。看看下面场景中主管与管理员的对话：

主管：小张，把我们公司的所有设备系统加入到监控系统中，我想接收所有的告警消息。

管理员：所有的设备，包括服务器？

主管：是的。

管理员：OK，没问题。

第二天。

主管：昨晚一夜没睡，手机短信响个不停。不是系统告警，就是暴力破解告警提示，要么就是扫描报警... ...

管理员：不是你说要看所有报警？

部署监控系统可没有你想象的那么容易，在上面的例子中，监控实施得不好，反而带来麻烦。目前，各种监控软件层出不穷，大家往往不知如何选择，Nagios 遵从 UNIX 理念“简单实用”，Nagios 本身已经模块化，包含了很多实用插件，Nagios 对所有插件状态进行跟踪，可以用来监视系统运行状态和网络信息。Nagios 可以监视所指定的本地或远程主机以及服务，同时提供异常通知功能。Nagios 可以提供以下几种监控功能。

- 监控网络服务（SMTP、POP3、HTTP 等）可用性。
- 监控主机资源（处理器负载、磁盘利用率等）参数增量变化。
- 简单的插件设计使得用户可以方便地扩展自己服务的检测方法。
- 当服务或主机问题产生可将告警发送给联系人（通过电子邮件、短信等方式）。
- 具备定义事件处理功能，可以在主机或服务的事件发生时获取更多问题定位。
- 可选的 Web 界面，用于查看当前的网络状态消息、通知故障等。

8.6.1 Nagios 实现原理

Nagios 通常由一个主程序（Nagios）、一个插件程序（Nagios-plugins）和 4 个可选的 ADDON（NRPE、NSCA、 NSClient++和 NDOUtils）组成。Nagios 的监控工作都是通过插件实现的，因此，Nagios 和 Nagios-plugins 是服务器端工作所必需的组件，可选组件功能如下：

（1）NRPE：用来在监控的远程 Linux/Unix 主机上执行脚本插件，以实现对这些主机资源的监控。

（2）NSCA：用来使被监控的远程 Linux/Unix 主机主动将监控信息发送给 Nagios 服务器。

（3）NSClient++：用来监控安装在 Windows 主机上的 Agent 组件。

（4）NDOUtils：用来将 Nagios 的配置信息和各种事件产生的数据存入数据库，以实现对这些数据的检索。

这 4 个附件中，NRPE 和 NSClient++工作于客户端，NDOUtils 工作于服务器端，而 NSCA 则需要同时安装在服务器端和客户端。

既然 Nagios 的功能是监控服务和主机，但是它自身并不包括这部分功能，所有的监控、检测功能都是通过各种类型插件来完成。当启动 Nagios 后，它会周期性地自动调用插件来检测服务器状态，同时 Nagios 会维持一个队列，所有插件返回来的状态信息都进入队列，Nagios 每次都从队首开始读取信息，并进行处理后，把状态结果通过 Web 显示出来。利用 Nagios 尤其适合监视大量服务器中的大批服务，但在绘图精细度上确实不及 Cacti，上一节讲解的 Ntop 应用就弥补了这一缺陷。

Nagios 提供的插件安装完成后，在 Nagios 主目录下的 libexec 里放有 Nagios 自带的可以使用的所有插件，如 check_disk 是检查磁盘空间的插件，check_load 是检查 CPU 负载的插件等。每一个插件可以通过运行./check_xxx -h 来查看其使用方法和功能。Nagios 可以识别 4 种状态返回信息：

- 0（OK）：表示状态正常。
- 1（WARNING）：表示警告，说明系统出现一定的异常。
- 2（CRITICAL）：表示出现非常严重的错误。
- 3（UNKNOWN）：表示未知，实际上被监控的对象已经停止或不可达。

8.6.2 利用 NRPE 插件实现服务器监控

当知道 Nagios 如何通过插件来管理服务器对象后，现在开始研究它是如何管理远端服务器对象。Nagios 系统提供了一个插件 NRPE。Nagios 通过周期性地运行它，来获得远端服务器的各种状态信息。它们之间的关系如图 8-52 所示。

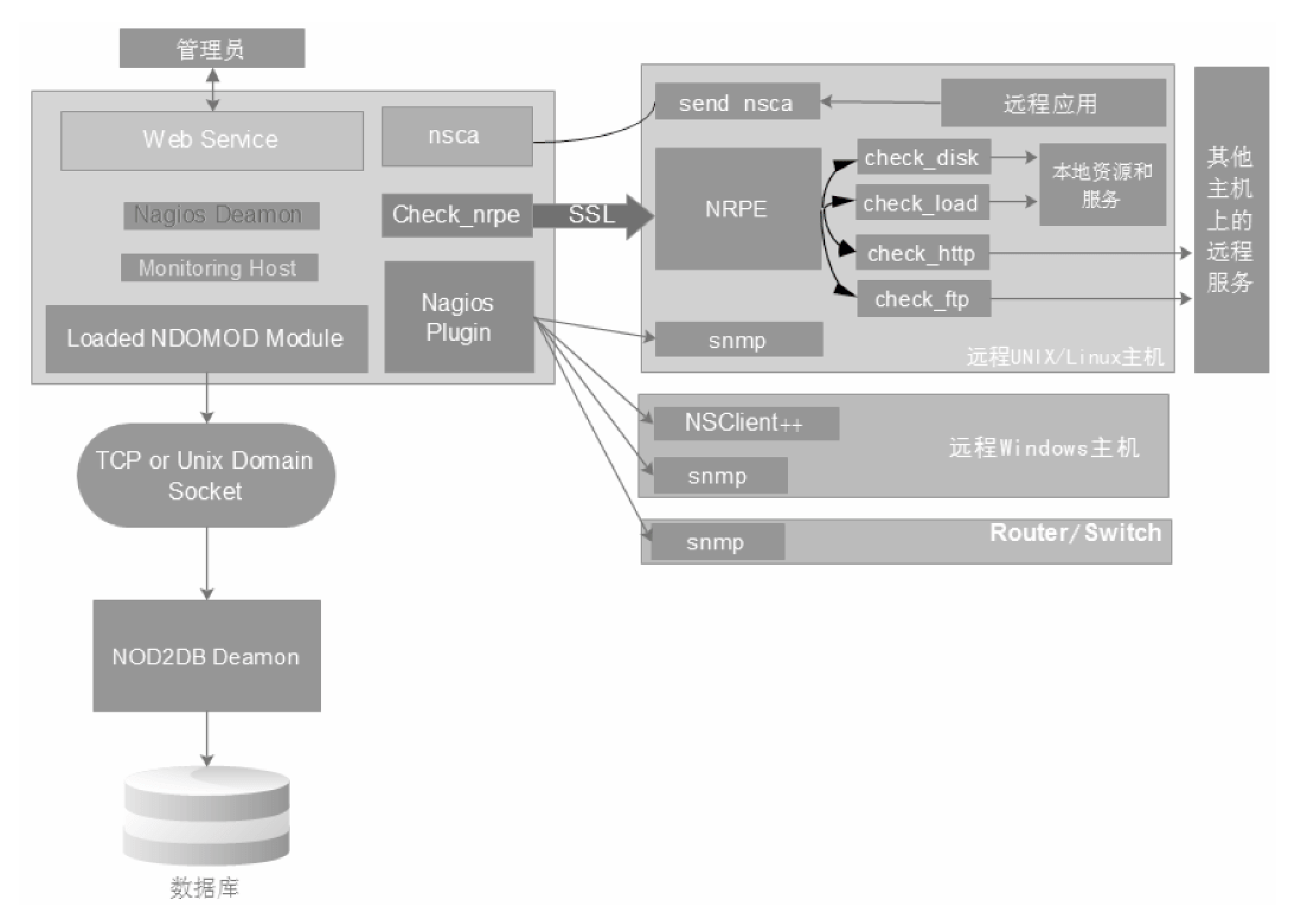


图 8-52　Nagios 架构图

Nagios 根据插件返回值来判断监控对象的状态，并通过 Web 显示出来，以供管理员及时发现故障。Nagios 最好用的地方就是它将这些每天管理员做的工作自动化，只需设定好要监听的端口即可，它会默默地工作，帮忙定时地去检测服务端口的状态，一旦发现问题，会及时发出报警。Nagios 的报表功能也很强大。管理员可以很容易获得到每天、每周和每月的 Service 运行状况。当有了 Nagios，哪怕就是管理上千台机器，也不会手忙脚乱，而会有一种运筹帷幄的感觉。下面我们来了解一下，如何在 OSSIM 系统中使用 Nagios 系统。

刚安装完 OSSIM 系统，并升级完成以后，就对监控网段的服务器或一些重要客户机进行扫描，首先选择右侧的 Assets→Asset Discovery 菜单，这时系统提示用户选择目标网段。注意，如果部署了多个 Sensor，那么嗅探对应网段需要和 Sensor 相对应（sensor 和待监控机器属于同一 VLAN 里）。比较合理的方式就是将服务器事先划分为不同的组，例如 Web 组、Ftp 组、Samba 组等。当扫描开始时系统后台调用 Nmap 工具开始扫描网络以收集数据，配置过程如图 8-53 所示。

图 8-53　添加监控网段主机

目标主机扫描完成，单击“Update database values”把数据更新到数据库中。如图 8-54 所示。

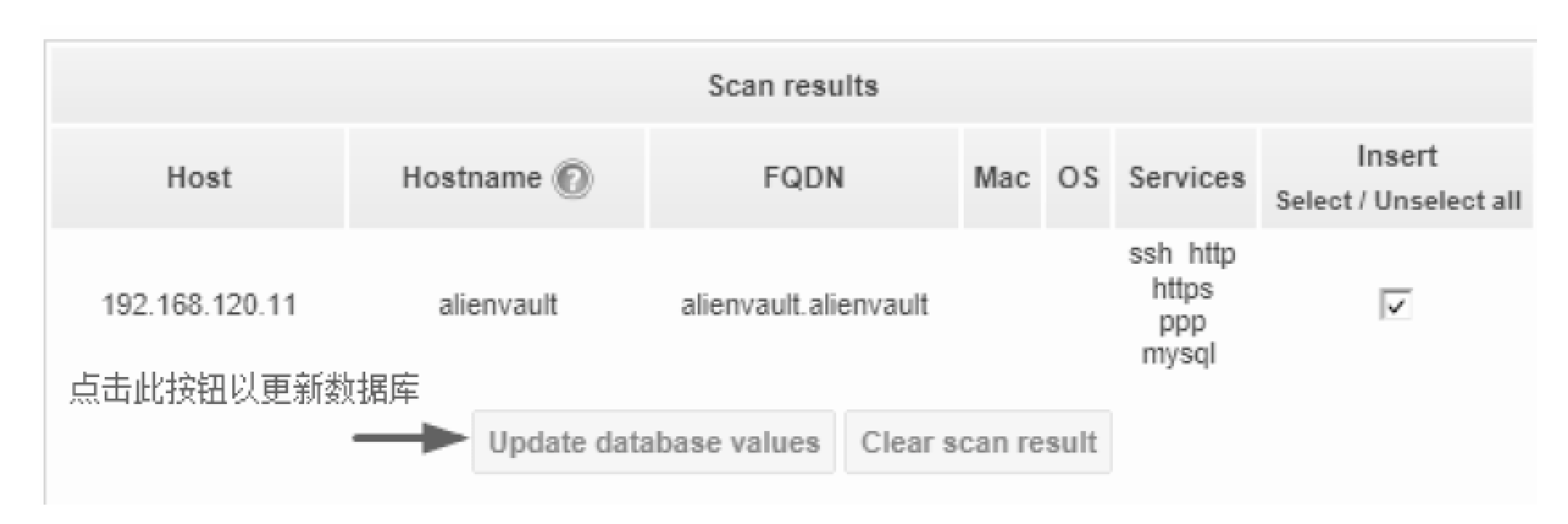


图 8-54　扫描主机结果

扫描过程中，为节约时间，建议关闭反向解析“Enable reverse DNS Resolution”，如图 8-55 所示。

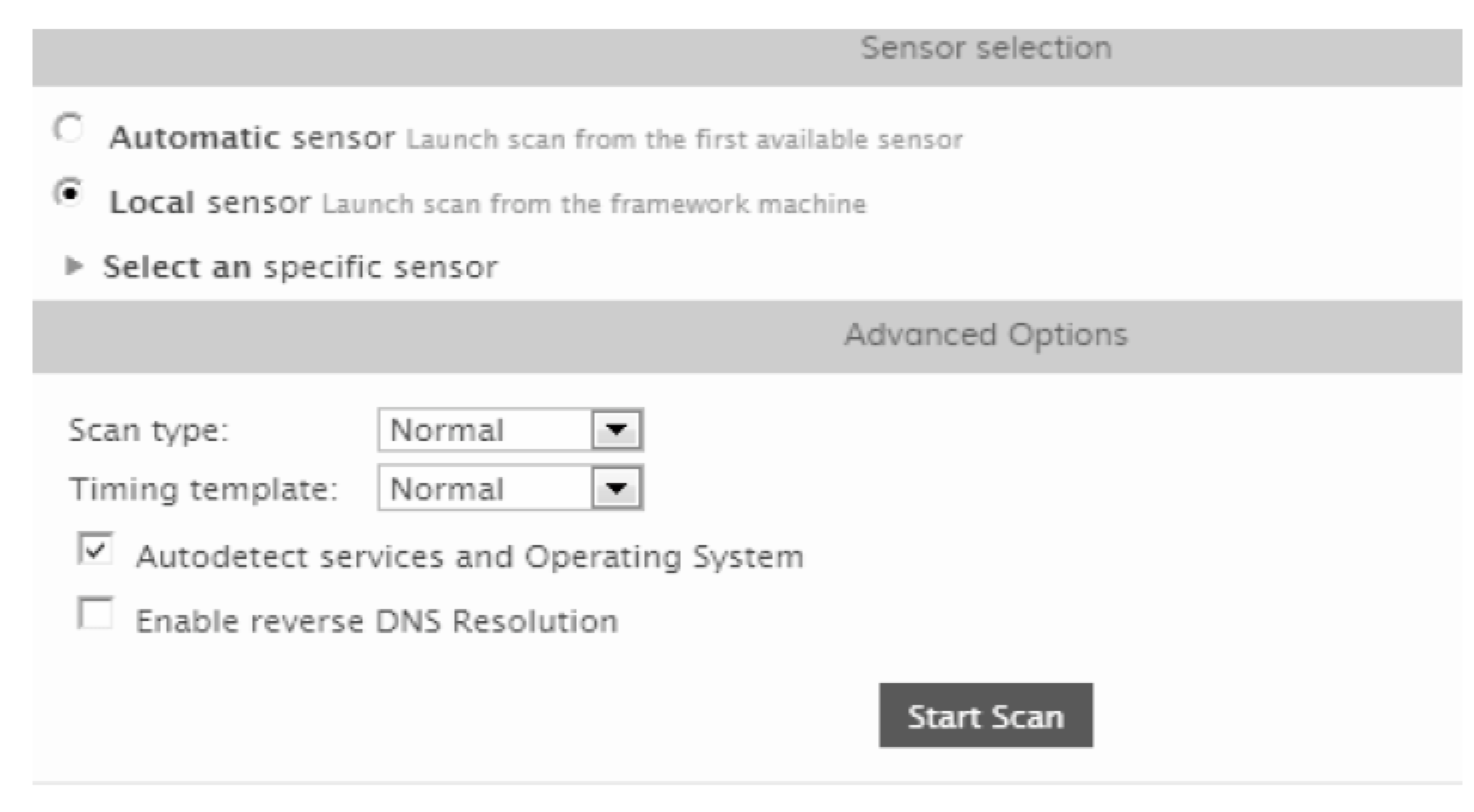


图 8-55　DNS 反向解析

单击“Start Scan”按钮，屏幕出现“Scanning network（10.32.14.0/24）with local Nmap, please wait...”，这时后台开始调用 nmap 命令开始扫描。

通过“ps –ef | grep nmap”命令可查看到命令执行的具体参数。

此时，系统列出当前网段内所有扫描到的主机列表，用户先检查是否和当前一致，然后选择某台主机，再选择修改按钮，系统弹出此主机配置信息，在配置界面可以选监控的设备属性以及服务。

8.6.3　Nagios 的 Web 界面

在 OSSIM 中打开 Nagios 的方法是 Environment→Availability，如图 8-56 所示。

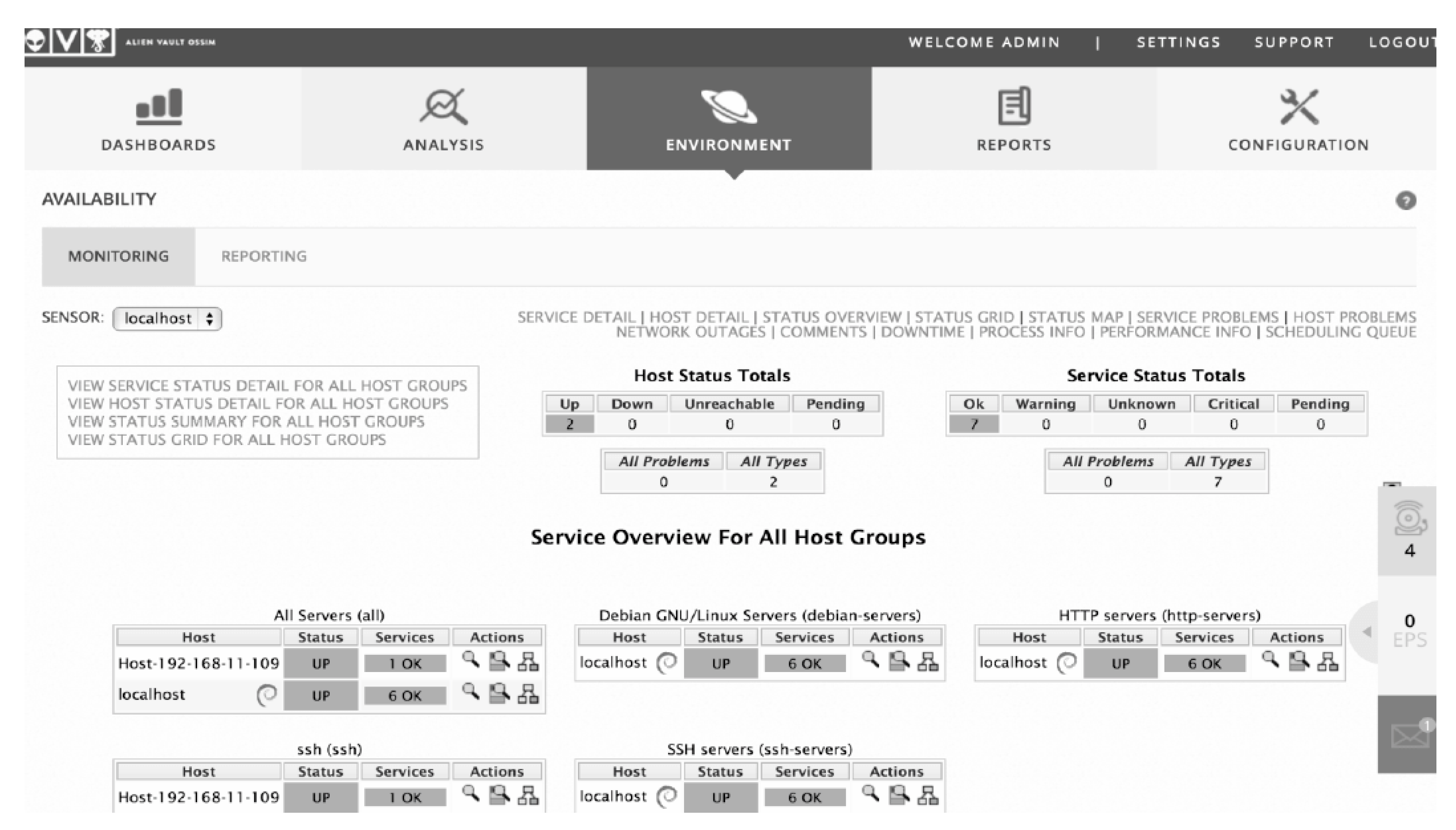


图 8-56　Nagios 主界面

这里大家看到的界面和传统 Nagios Web 界面有些不同，这里省去了配置菜单。这个 Web 界面是基于 CGI 实现，所以运行速度很快，读者可以在 Ossim 系统的/usr/lib/cgi-bin/nagios3/目录下查看所有程序（已加密）。这些 CGI 脚本，从 Nagios 所维护的 var 目录下的多个日志文件中收集所需要的信息。因为完整的 Web 界面够写一本书了，所以下面重点展示重要的 Web 视图，这些指标可以反映主机资源可用性。

1. Monitoring（当前监控状态）

- Service Detail，服务细节。
- Host Detail，主机细节。
- Status Overview，状态概览。
- Status Grid，网格状态。

- Status Map，状态地图。
- Service Problems，服务问题。
- Host Problems，主机问题。
- Network Outages，网络中断。
- Comments，注释。
- Downtime，宕机时间。
- Process Info，进程信息。
- Performance info，性能信息。
- Scheduling Queue，计划队列。

状态视图的后台程序主要是/usr/lib/cgi-bin/nagios3/status.cgi，Nagios 大部分功能都要与这个 CGI 文件交互。下面单击“Service Detail”超链接，如图 8-57 所示。

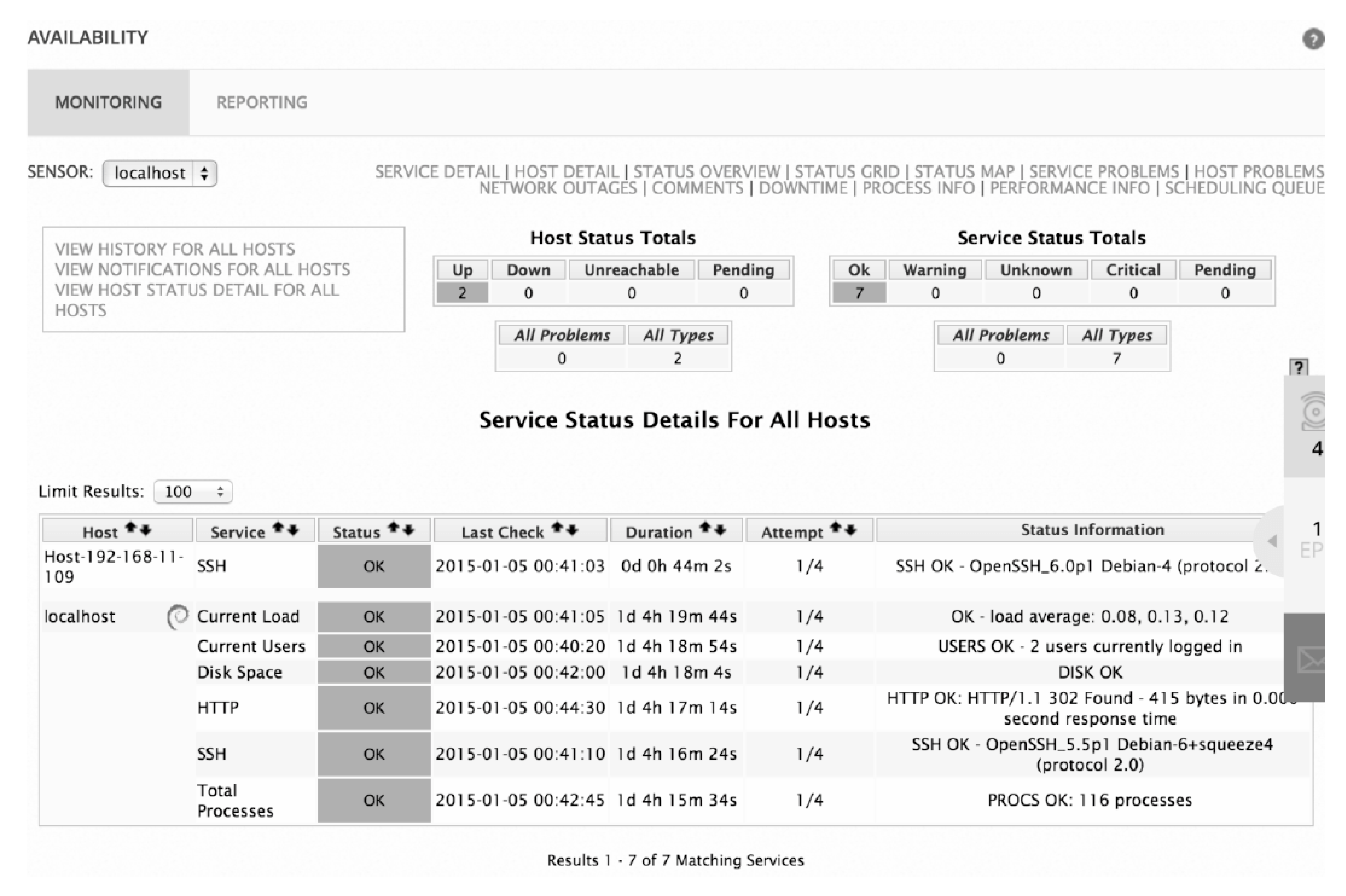


图 8-57　显示服务详情

单击上面这些菜单会发现，每次打开的界面有个共同点，前七个菜单，它们显示了四个部分：

- 基于上一个链接的子菜单，提供快捷链接。
- 主机状态表（Host Status Totals），展示区域中的主机状态。
- 服务状态表（Service Status Totals），展示区域中的服务状态。
- 细节展示区域。

下面我们单击图中主机 127.0.0.1，如图 8-58 所示。

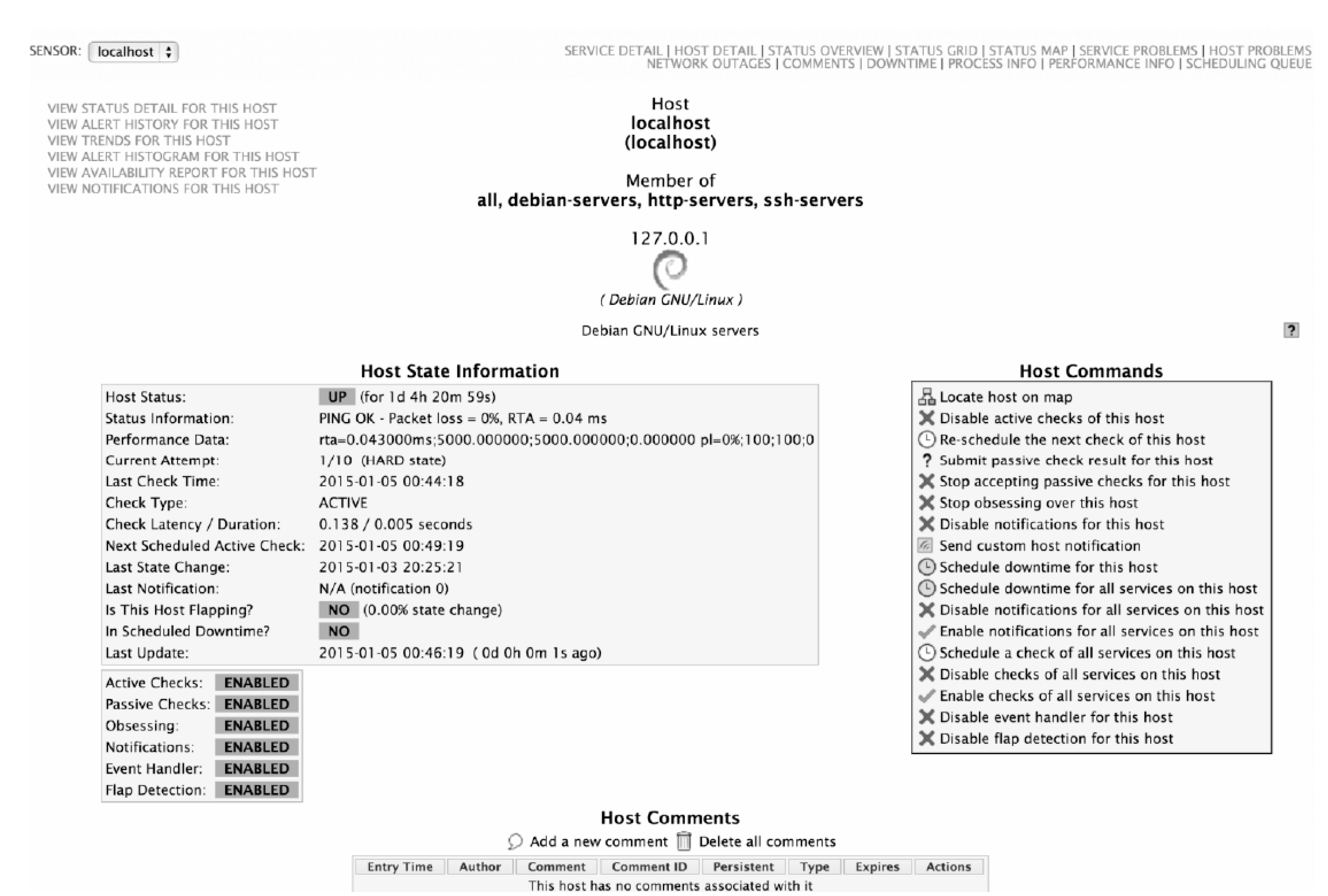


图 8-58　显示主机详情

该图显示的是某台主机的详细信息，大家可以看到，只有主机状态信息，服务端口信息并没有直接显示出来，但在上图左侧显示了一个详细的菜单：

- VIEW STATUS DETAIL FOR THIS HOST
- VIEW ALERT HISTORY FOR THIS HOST
- VIEW TRENDS FOR THIS HOST
- VIEW ALERT HISTOGRAM FOR THIS HOST
- VIEW AVAILABILITY REPORT FOR THIS HOST
- VIEW NOTIFICATIONS FOR THIS HOST

以上这些菜单可以显示主机详细状态。主机状态细节展示 Host Status Detail，如图 8-59 所示。

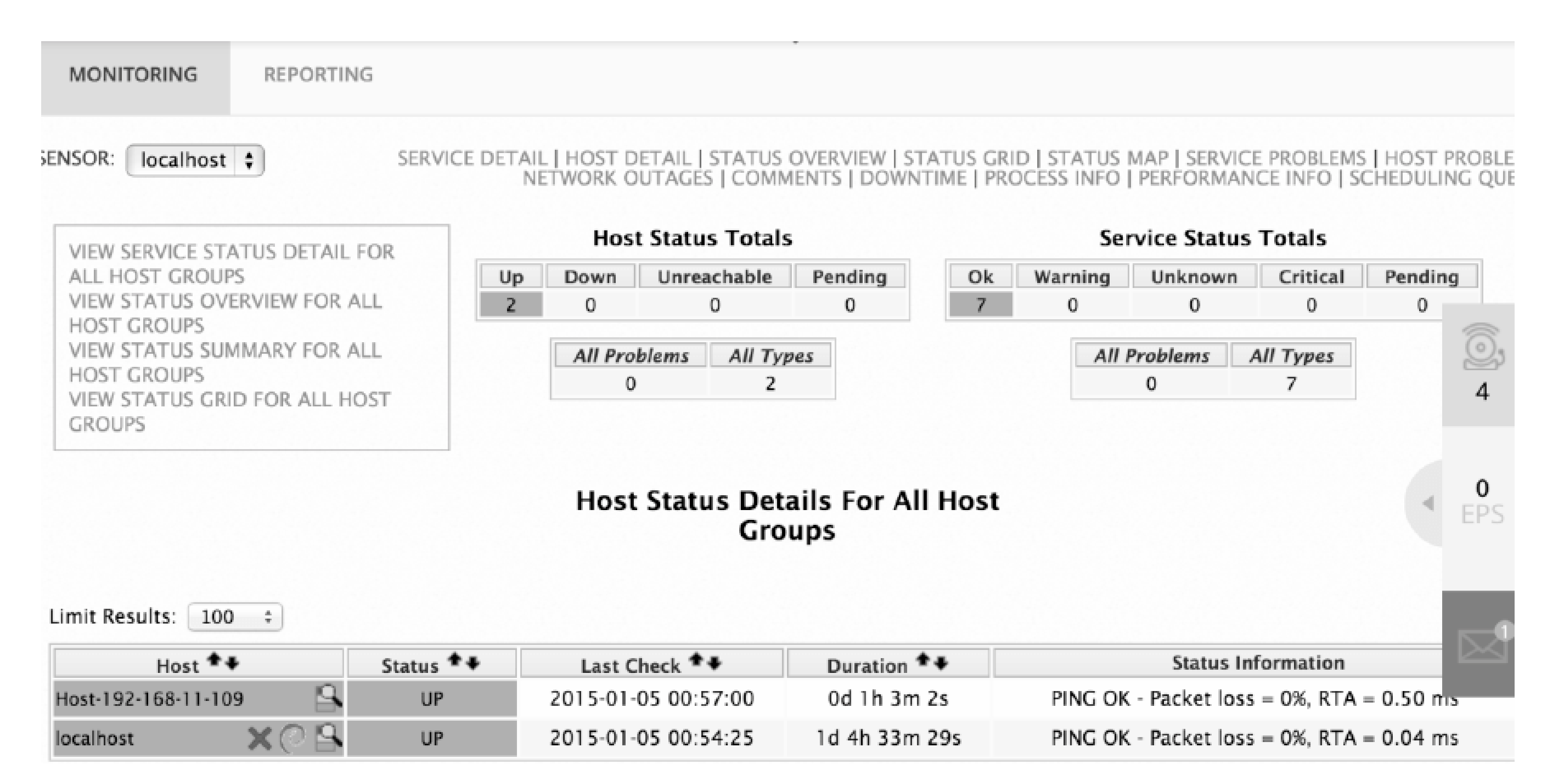


图 8-59　主机细节

选择页面右上侧的 View Status Overview for all host groups 和 View Host Status detail for all host groups 菜单，这两个菜单非常实用，它在一屏内，按主机组分组，显示了监控环境整体状态。如图 8-60 所示。当管理的主机数量达到上百台时，它集中展示了所有主机和服务的状态，总体展示当前环境。

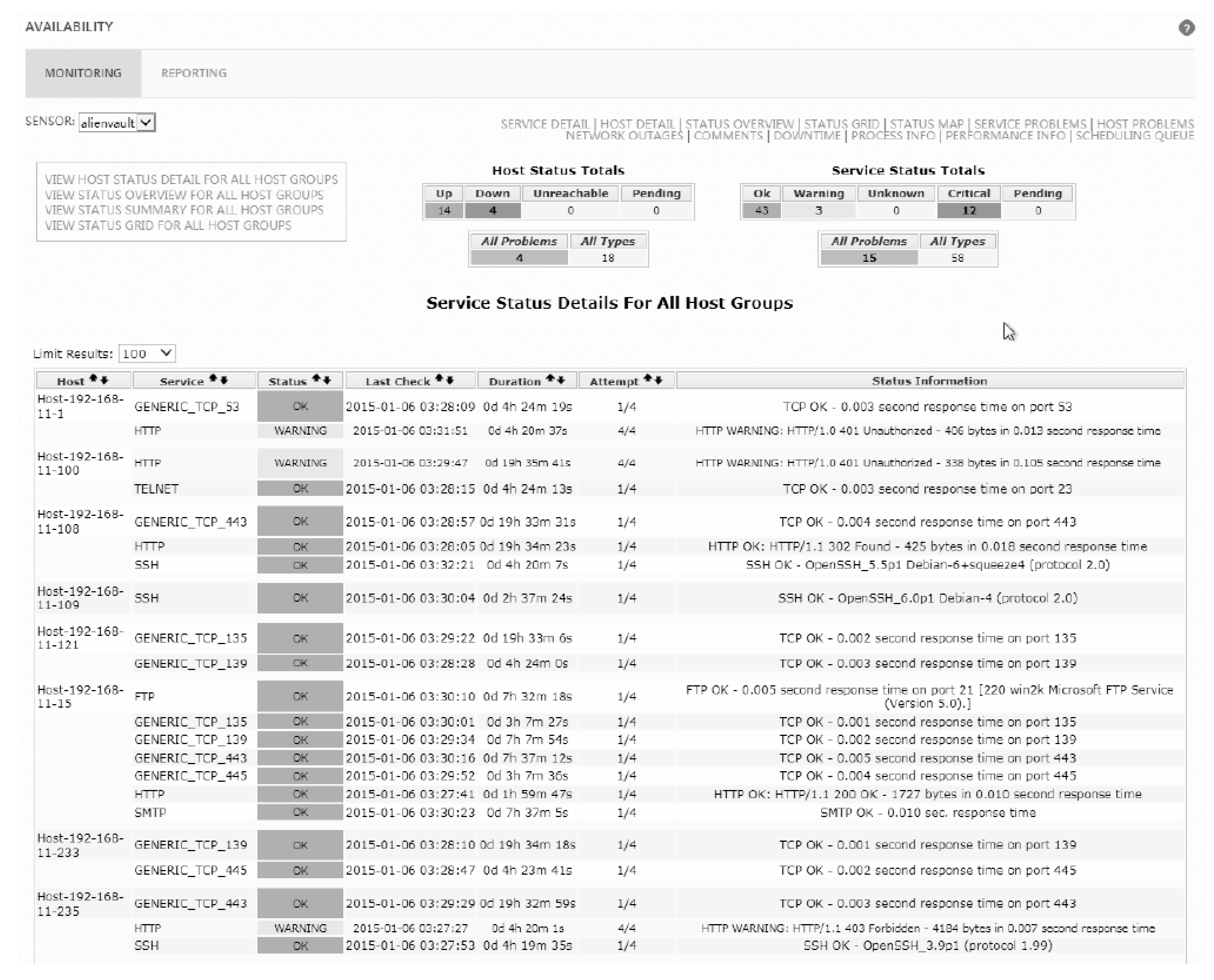


图 8-60　所有主机组预览

2. Reporting（报表）

Nagios 中内置了丰富的报表功能，负责提供报表功能的程序也是 CGI 程序。主要分为以下几种：

- Trends，趋势图（能够直观地再现服务状态与时间之间的关系）。
- Availability，可用性。
- Event Histogram，事件直方图（或柱状图）。
- Event History，事件历史。
- Event Summary，事件汇总。
- Notifications，通知。
- Performance info，性能日志。

单击相应报表类型，系统会提示指定对象类型，以生成相应的报表。例如，我们单击可用性报表，观察趋势图。首先选择 Trends 按钮，如图 8-61 所示。

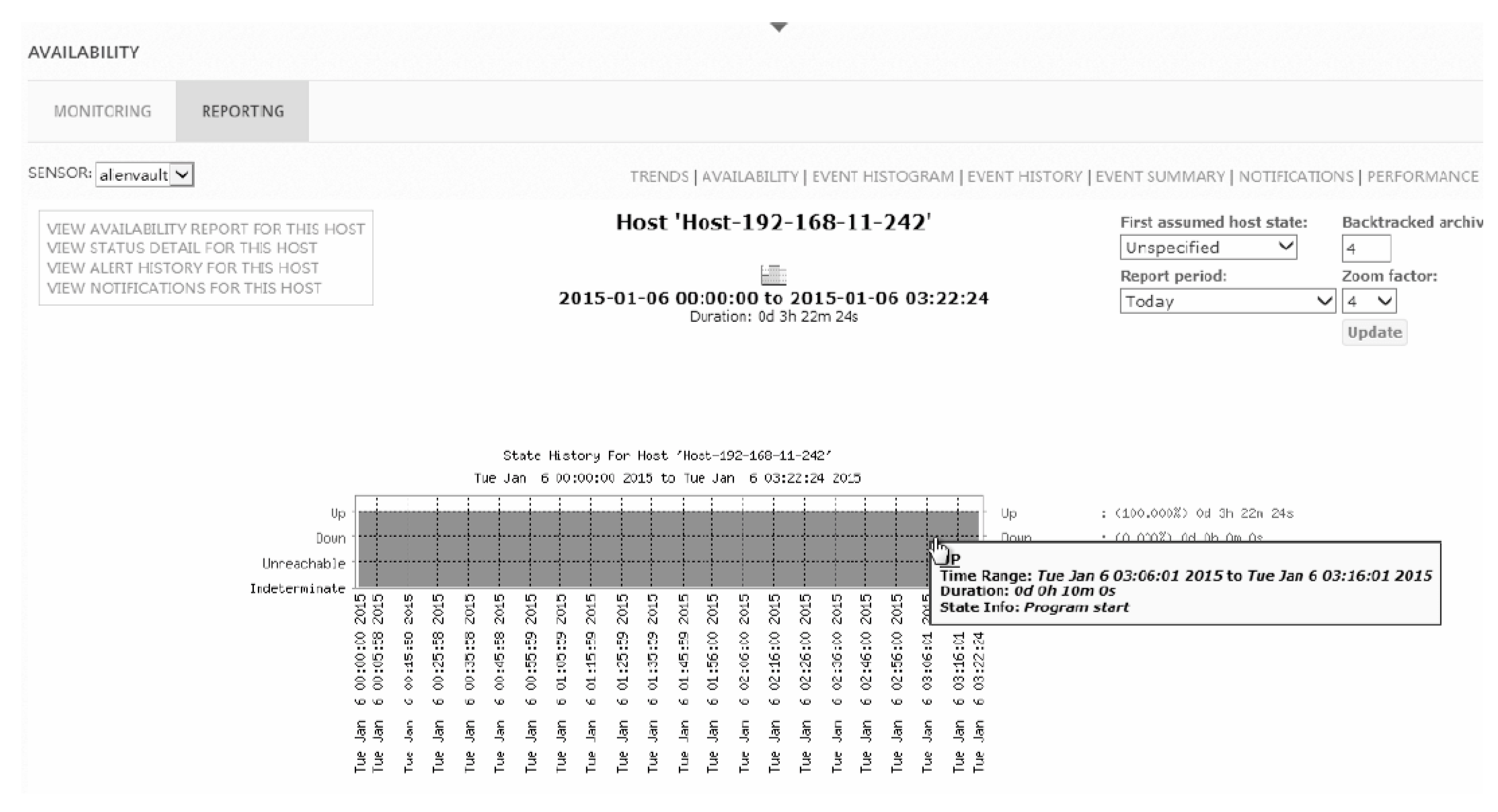


图 8-61　趋势图

大家观察趋势图时，主要看 X 轴，在该图中 X 轴表示时间，而 Y 轴则表示服务状态（正常、警告、未知、严重）。

直方图的设置方法大致和趋势图一样，直方图和趋势图的区别就在 X 轴上。假如报表的周期为 7 天，趋势图中的 X 轴对应的就是 7 天中的每一天，Y 轴就是服务状态（例如正常、警告、严重、未知），在该图中记录了服务状态和时间的关系，如图 8-62 所示。

而直方图在 X 轴上显示用户自定义的故障统计的时间间隔，比如每天的小时数，每周的天数。在 Y 轴显示故障状态的数量。例如，选择周期为最近 7 天，故障统计的时间间隔为每

天 24 小时，那么 X 轴显示 24 个小格，每格代表 1 小时，Y 轴会显示故障状态在每个小时发生的次数总和。这样表示更有助于通过可视化展现服务区故障的发展趋势。比如，服务器 A 的 CPU 利用率很高，那么在直方图上就能展现出每周，某一个时间（晚上 12 点）服务器 A 的 CPU 利用率。

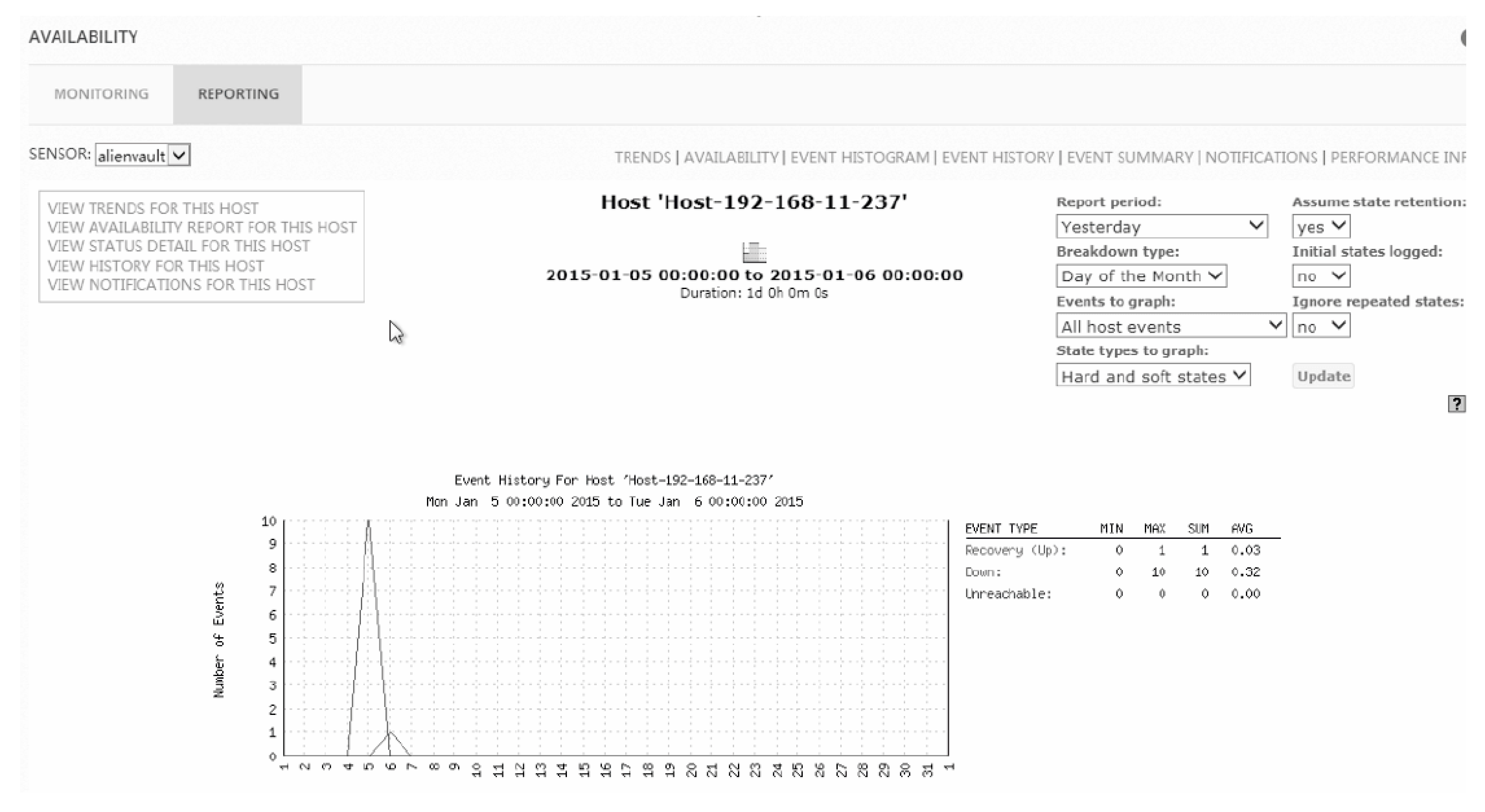


图 8-62　事件直方图

8.6.4　Naigos 插件

Nagios 的插件才是真英雄，没有 Nagios 如同一个空壳，插件的工作步骤如下：

- 从被监控主机上收集信息。
- 将收集信息与阈值进行比较。
- 根据比较结果，提供对应退出返回码（0、1、2、3）。

1. 理解插件

插件如何而来，我们先来看个最简单的例子。编写一个 ping 脚本，它的功能是向服务器 192.168.11.108（也可以使用 hostname 的形式）发送 5 个 ICMP 包，与使用 ping 命令不同的是，这个脚本提供了退出代码，如果成功 ping 通，则退出代码为 0，如果 ping 不通，那么表示有故障，退出代码为 2。脚本内容如图 8-63 所示。

```
#!/bin/sh
if ping -qc 5 192.168.11.108
then
    exit 0
else
    exit 2
fi
```

图 8-63　ping 脚本

不过输出 0、2，只有程序员知道什么意思，对于不懂得什么是退出代码的用户来说没有帮助，下面继续优化这个脚本，通过标准输出提供一行可读信息。如图 8-64 所示。

```
#!/bin/sh
OUTPUT=`ping -c5 192.168.11.108 | tail -n2`
if [ $? -gt 0 ]
then
    echo "CRITICAL!!  $OUTPUT"
    exit 2
else
    echo "OK!  $OUTPUT"
    exit 0
fi
```

图 8-64　改进型 ping 脚本

在该脚本中，使用 OUTPUT 变量，搭配 tail 命令来获取 ping 命令输出的最后两行信息，这个脚本执行结果如图 8-65 所示。

```
localhost:~# ls
ping.sh
localhost:~# ./ping.sh
OK!  5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.023/0.042/0.059/0.012 ms
```

图 8-65　脚本执行

此时，大家是不是对插件的作用有所体会？在 Nagios 的架构工作方式中，最大的好处是，能够通过命令行执行插件，非常方便扩展，因为每个插件都是一个独立的小脚本。以 OSSIM 为例，在/usr/lib/nagios/plugins/目录下存有大量插件，每个插件还可以调用帮助信息，例如：

```
#./check_ping -h
```

在继续下面的命令之前，先确保 OSSIM 能连接互联网，测试命令如图 8-66 所示。

```
#./check_ping -H 8.8.8.8 -w 200,20% -c 300,50% -p 3 -t 2
```

```
alienvault:/usr/lib/nagios/plugins# ./check_ping -H 8.8.8.8 -w 200,20% -c 300,50
% -p 3 -t 2
PING WARNING - Packet loss = 33%, RTA = 243.30 ms|rta=243.304001ms;200.000000;30
0.000000;0.000000 pl=33%;20;50;0
alienvault:/usr/lib/nagios/plugins# _
```

图 8-66　插件执行

各参数含义如下：

- -H：主机地址。

- -w：WARNING 警告状态：响应时间（毫秒），丢包率（%），阈值。
- -c：CRITICAL 危险状态：响应时间（毫秒），丢包率（%），阈值。
- -p：发送的包数，默认 5 个包。
- -t：超时时间，默认 10 秒。

这样可以是操作者对插件问题进行故障分析。如果 Nagios 在调用某个插件时出现了问题，我们可以在 shell 提示符下直接运行插件，并加入相同参数，查看具体发生了什么问题。大家如果需要了解插件问题，还可以参考 https://nagios-plugins.org/doc/guidelines.html。

在 OSSIM 系统中已安装好 naigos-plugins-1.4.15，在/usr/lib/naigos/plugins/目录下有很多 check 开头的二进制文件，它们即为 Nagios 插件，部分插件如图 8-67 所示。

```
alienvault:/usr/lib/nagios/plugins# ls
check_apt        check_flexlm        check_jabber       check_nntp       check_radius      check_time
check_bgpstate   check_fping         check_ldap         check_nntps      check_real        check_udp
check_breeze     check_ftp           check_ldaps        check_nt         check_rpc         check_ups
check_by_ssh     check_game          check_linux_raid   check_ntp        check_rta_multi   check_users
check_clamd      check_host          check_load         check_ntp_peer   check_sensors     check_wave
check_cluster    check_hpjd          check_log          check_ntp_time   check_simap       negate
check_dhcp       check_http          check_mailq        check_nwstat     check_smtp        urlize
check_dig        check_icmp          check_mrtg         check_oracle     check_snmp        utils.pm
check_disk       check_ide_smart     check_mrtgtraf     check_overcr     check_spop        utils.sh
check_disk_smb   check_ifoperstatus  check_mssql        check_pgsql      check_ssh
check_dns        check_ifstatus      check_mysql        check_ping       check_ssmtp
check_dummy      check_imap          check_mysql_query  check_pop        check_swap
check_file_age   check_ircd          check_nagios       check_procs      check_tcp
```

图 8-67　Nagios 插件

系统并没有提供每个监控程序的脚本的说明文档，想了解这些脚本如何工作的话，可以通过--h 参数，显示其使用方法和参数。例如：#./check_icmp –h，根据提示的方法，可以在命令行中尝试使用以下命令（如图 8-68 所示）：

```
alienvault:/usr/lib/nagios/plugins# ./check_icmp -H sina.com
OK - sina.com: rta 151.072ms, lost 0%|rta=151.072ms;200.000;500.000;0; pl=0%;40;80;; rtmax=152.519ms
;;;; rtmin=150.319ms;;;;
alienvault:/usr/lib/nagios/plugins#
```

图 8-68　插件帮助信息

从返回结果可以看到状态值“OK”，以及一些详细的数据信息。

下面我们来看看其他插件功能的例子：

（1）Check_load 检测负载，通过系统命令 top 显示，check_load 并不是 CPU 的负载，check_load 是检查系统正在运行的任务数+等待的任务数，如图 8-69 所示。

```
alienvault:/usr/lib/nagios/plugins# ./check_load -w 15,10,5 -c 30,25,20
OK - load average: 1.74, 0.67, 0.48|load1=1.740;15.000;30.000;0; load5=0.670;10.
000;25.000;0; load15=0.480;5.000;20.000;0;
alienvault:/usr/lib/nagios/plugins# _
```

图 8-69　检测负载

当 1 分钟多于 15 个进程等待，5 分钟多于 10 个，15 分钟多于 5 个，则为 warning 状态。

当 1 分钟多于 30 个进程等待，5 分钟多于 25 个，15 分钟多于 20 个，则为 critical 状态。

（2）Check_swap 检测交换分区，如图 8-70 所示。

```
alienvault:/usr/lib/nagios/plugins# ./check_swap -w 20% -c 10%
SWAP OK - 100% free (865 MB out of 865 MB) |swap=865MB;173;86;0;865
alienvault:/usr/lib/nagios/plugins#
```

图 8-70 检测交换分区

（3）检查内存

当使用 NRPE 时，这条命令会检测空闲的内存，当可用内存小于 20%时会发出警告，并且在可用内存小于 10%时会生成一个严重警告。在 Nagios 监控中，默认是没有 check_mem 这个插件，为了完成下面的实验，首先到我的博客下载 check_mem.pl 脚本（如图 8-71 所示）。

```
alienvault:/usr/lib/nagios/plugins# ls -l check_mem.pl
-rwxrwxrwx 1 root root 4138 Apr  2 04:25 check_mem.pl
alienvault:/usr/lib/nagios/plugins# mv check_mem.pl check_mem
alienvault:/usr/lib/nagios/plugins# ./check_mem -f -w 20 -c 10
Memory CRITICAL - 8.5% (239544 kB) free
alienvault:/usr/lib/nagios/plugins# _
```

图 8-71 检查内存

如果你看到类似上面那样的输出，则表示该命令正常，接下来需要定义 NRPE 检查内存使用率的命令。

```
#vi /etc/nagios/nrpe.cfg
```

加入以下内容：

```
command[check_mem]=/usr/lib/nagios/plugins/check_mem -f -w 20 -c 10
```

（4）检测 FTP 服务，效果如图 8-72 所示。

```
alienvault:/usr/lib/nagios/plugins# ./check_ftp -H localhost -p 21
FTP OK - 0.002 second response time on port 21 [220 (vsFTPd 2.3.2)]|time=0.00203
1s;;;0.000000;10.000000
alienvault:/usr/lib/nagios/plugins#
```

图 8-72 检测服务

2. 远程获取信息

在该实验中，我们通过 SSH 获取远程主机的负载信息，首先要把密钥对中的公钥，复制到要访问的机器中，并保存为~/.ssh/authorized_keys。该实验可以形象理解 Nagios 在远程服务器上运行插件的情形。下面我们要查询远程系统的负载平均值，由于做了上面的准备工作，我们输入以下命令（将该命令保存文件名为 load_check.sh）：

```
#ssh  192.168.11.108 "uptime |cut -d: -f5"
```

此时会在远程服务器 192.168.11.108 上，输出 1 分钟、5 分钟、15 分钟的平均负载信息。当把这个脚本存放在远程服务器的/usr/local/bin/目录后，管理员就可以通过 SSH 远程支持，类

似如下命令：

```
#ssh  192.168.11.108 "/usr/local/bin/load_check.sh"
```

实际上管理员还会通过 cron 命令，在远程服务器 192.168.11.108 上调度计划任务定期执行。将上面这行命令稍加变形，加入退出码，就变成如图 8-73 所示的脚本。

```
#!/bin/sh
LOAD=`uptime |awk '{print $10}'`
if [ $? -gt 1 ]
then
     echo "Critical ! load on `hostname` is $LOAD"
     exit 2
else
     echo "OK! Load on `hostname` is $LOAD"
     exit 0
fi
```

图 8-73　脚本

开始检验成果，这时考虑 Nagios 如何执行远端命令呢？难道只能在本地执行？答案很简单，只需要使用 SSH 就可以搞定。这种方式就是 Nagios 远程执行的基础。但由于认证等问题，以上介绍的 SSH 的方式，从技术上可以实现，但从安全角度并不理想，所以就用到了 NRPE（远程插件执行程序），NRPE 是完全跨平台的，它可以运行在 Windows 和 UNIX 平台上。

安装好的每个插件都是可以独立使用，那么在 Nagios 中，如何调用这些插件的呢？如果要加入参数，需要用哪种格式呢？首先，要了解这些插件会被 Nagios 用在什么地方。Nagios 有很多个 cfg 文件，用来定义各式各样的信息，其中 hosts.cfg 和 services.cfg（一般是这两个，也可能是其他定义主机和服务的配置文件）是用来定义主机和服务信息。这些插件就被使用在这里。例如，在 OSSIM 系统中服务监控定义 nagios 在/etc/nagios3/conf.d/ossim-configs/host-services/alienvault-alienvaultSSH.cfg（注意：alienvault-alienvault 为设定的主机名）配置文件中，如图 8-74 所示。

```
alienvault:/etc/nagios3/conf.d/ossim-configs/host-services# cat alienvault-alienvaultSSH.cfg
define service{
        host_name alienvault-alienvault
        service_description SSH
        check_command check_ssh
        use generic-service
        notification_interval 120
        }
alienvault:/etc/nagios3/conf.d/ossim-configs/host-services# ls
alienvault-alienvaultGENERIC_TCP_3000.cfg  alienvault-alienvaultGENERIC_TCP_8080.cfg
alienvault-alienvaultGENERIC_TCP_443.cfg   alienvault-alienvaultSSH.cfg
alienvault-alienvaultGENERIC_TCP_80.cfg
```

图 8-74　cfg 配置文件

Host_name 项说明该服务所在的主机名，service_descrition 项为说明信息，这项内容会显示在 Nagios 页面上，所以尽量简洁明了。这个服务定义了 naigos 需要监控的内容和手段，以及用 check_ssh 插件来监控主机 alienvault-alienvualt 上的 SSH 服务。

3. 主机监控

Nagios 的主要功能是监控，其监控对象包括主机和服务。针对主机监控的配置项都是怎样的呢？在 OSSIM 系统中，对于主机监控信息存放在/etc/nagios3/conf.d/ossim-configs/hosts 目录下，文件特征是 ip.cfg，如图 8-75 所示。

```
alienvault:/etc/nagios3/conf.d/ossim-configs/hosts# ls
10.32.14.128.cfg  10.32.14.131.cfg  10.32.14.155.cfg  10.32.14.21.cfg   192.168.56.28.cfg
10.32.14.130.cfg  10.32.14.133.cfg  10.32.14.20.cfg   10.32.14.254.cfg
alienvault:/etc/nagios3/conf.d/ossim-configs/hosts# cat 10.32.14.133.cfg
define host{
        host_name alienvault-alienvault
        alias alienvault-alienvault
        address 10.32.14.133
        use generic-host
        }
alienvault:/etc/nagios3/conf.d/ossim-configs/hosts#
```

图 8-75　主机监控配置文件

IP 代表监控主机的 IP 地址，如例子中的 10.32.14.128、10.32.14.130 等。alias 表示别名，可以更详细地说明主机。address 表示 IP 地址。

8.6.5　Nagios 扩展 NRPE

NRPE 是监控软件 Nagios 的一个扩展，用于被监控的服务器中，向 Nagios 监控平台提供该服务器的一些本地的情况。例如，CPU、RAM、Disk 等等。NRPE 可以理解为 Nagios 的 for Linux 客户端。下面 Sensor 充当监控端服务器。注意一个问题，因为 NRPE 的通信端口为 TCP 5666，所以确保主机上这个端口没有被占用。监控端服务器（被检测主机）安装 NRPE 执行以下 3 条指令：

```
#apt-get install nagios-nrpe-server
#apt-get install nagios-nrpe-plugin
#apt-get install libnagios-plugin-perl
```

本机测试：

```
#nrpe -c /etc/Nagios/nrpe.cfg -d              \\如没有任何提示表示配置正确
```

接着开始 check_load 测试，我们利用 NRPE 这一工具，如图 8-76 所示。

```
alienvault:/usr/lib/nagios/plugins# nrpe -c /etc/nagios/nrpe.cfg  -d
alienvault:/usr/lib/nagios/plugins# /usr/lib/nagios/plugins/check_nrpe -H localh
ost -c check_load
OK - load average: 0.46, 0.65, 0.52|load1=0.460;15.000;30.000;0; load5=0.650;10.
000;25.000;0; load15=0.520;5.000;20.000;0;
alienvault:/usr/lib/nagios/plugins# _
```

图 8-76　nrpe 测试

待调试完毕，在远程主机上测试，不过需要修改/etc/nagios/nrpe.cfg 文件，在 allowed_hosts=127.0.0.1 之后，加上远程主机的 IP 地址，如图 8-77 所示。

```
#/etc/init.d/nagios-nrpe-server     \\重启服务
```

```
alienvault:/usr/lib/nagios/plugins# ./check_nrpe -H 192.168.91.222  -c check_load
OK - load average: 0.23, 0.47, 0.47|load1=0.230;15.000;30.000;0; load5=0.470;10.000;25.000;0; load15=0.470;5.000;20.000;0;
alienvault:/usr/lib/nagios/plugins# _
```

图 8-77　执行 check_nrpe

这样就不会报“Could not complete SSL handshake”错误。

OSSIM Server 添加配置：

```
#nano /etc/nagios/nrpe.cfg
command[check_disk]=/usr/lib/nagios/plugins/check_disk -w 20 -c 10
```

重启 NRPE 服务：

```
#/etc/init.d/nagios-nrpe-server restart
```

8.6.6　监控开销

首先，Nagios 监控系统自然也会消耗一定的开销，主要被监控主机上的 CPU 资源消耗和网络流量的消耗。Nagios 将服务检测输出到插件中。希望监控系统能够采用集中式的执行方式，因为那样被监控主机的资源占用就会更少。而且所有配置文件也能保存在一个地方。事实上对于少量服务器监控，这种想法完全正确，但如果管理 1 万台服务器，那么监控服务器的负担则会相当繁重。

采用集中模式就不合适。这种大型环境中，监控服务器就依赖被监控主机进行服务自行检测，然后把结果返回给监控服务器，有些类型的检测，没有必要在中心服务器上执行，例如检测 CPU、内存以及磁盘空间等信息。

其次就是带宽的开销。所有插件只要向中心服务器上报数据，即会产生流量，所以需要精简插件避免冗余插件的存在，另外，还需要减少监控主机的第三层路由流量。出于资源的管理的考虑，我们需要对资源进行逻辑分组，OSSIM 也为监控系统提供了 Hostgroup（主机组）的模块。

最后，使用主机组可以通过对配置进行优化，以满足该组的需要，比如，监控远程办公室的服务器的超时时间，可以设定得高一些，而本地服务器组的超时时间则可降低一些，从而减小报警的数量。

8.6.7　OSSIM 系统中应用 Nagios 监控资源

使用 OSSIM 监控资产的优势就是它集成了 Nagios+OCS 这样的资产监控工具。不仅如此，在 OSSIM 中设置监控的过程也非常人性化。下面介绍设置的步骤，首先，在资产列表栏中选中某个设备，进行如下几步操作。

（1）选择“EDIT AVAILABILITY MONITORING”编辑可用性监控按钮，如图 8-78 所示。

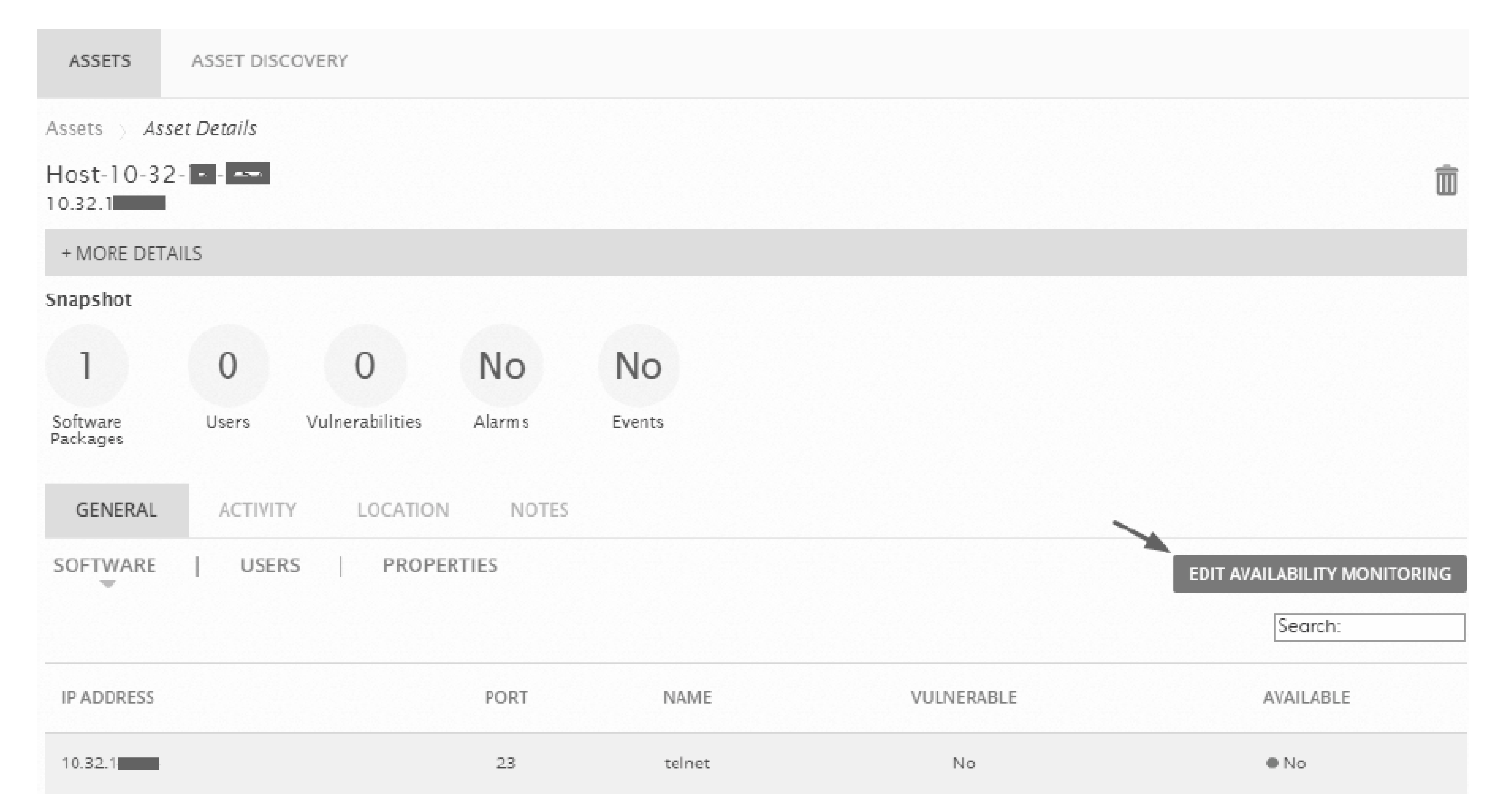


图 8-78　编辑可用性监控按钮

（2）选择监控端口后，单击“TOGGLE AVAILABILITY MONITORING”按钮，如图 8-79 所示。

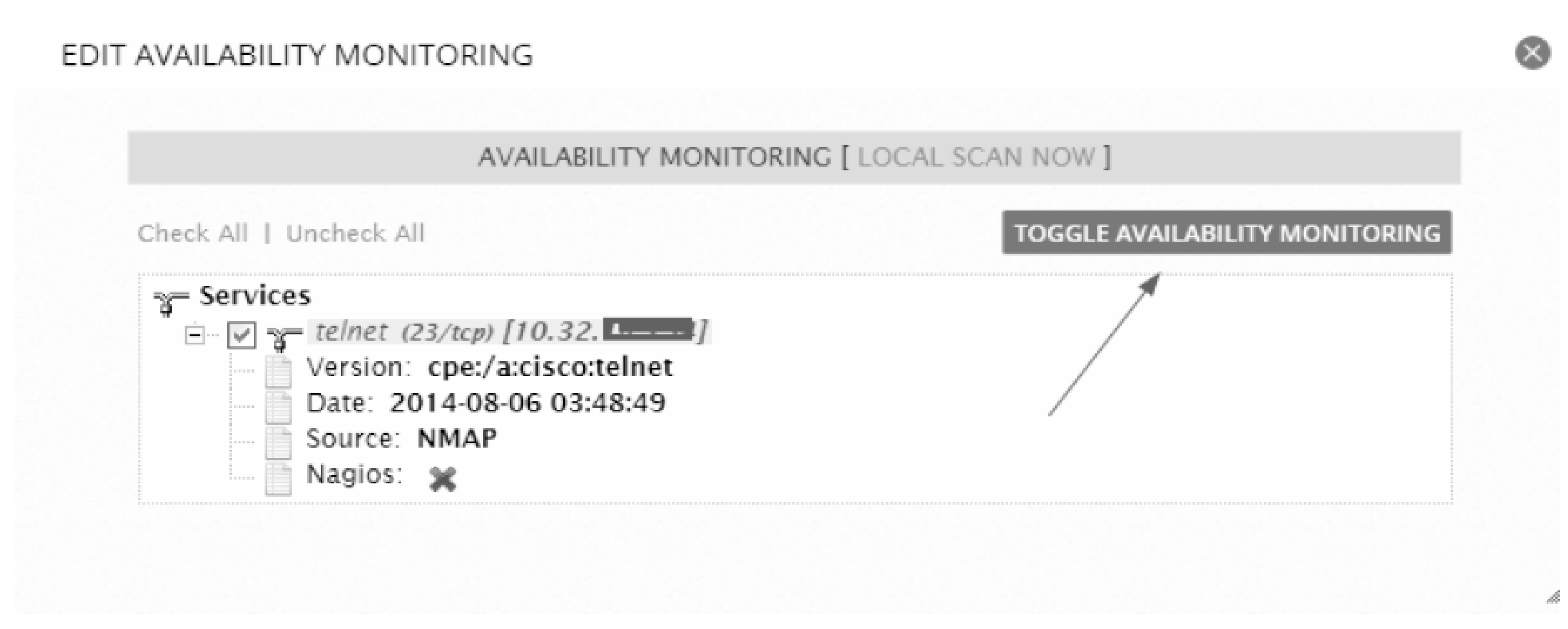


图 8-79　选择监控端口

EDIT AVAILABILITY MONITORING

Host services toggled successfully

如果成功选择，系统提示添加主机服务成功。

此时，该设备的可用性将被启用，同时在 Availability Monitoring 处绿色的圆形图标被点亮，如图 8-80 所示。

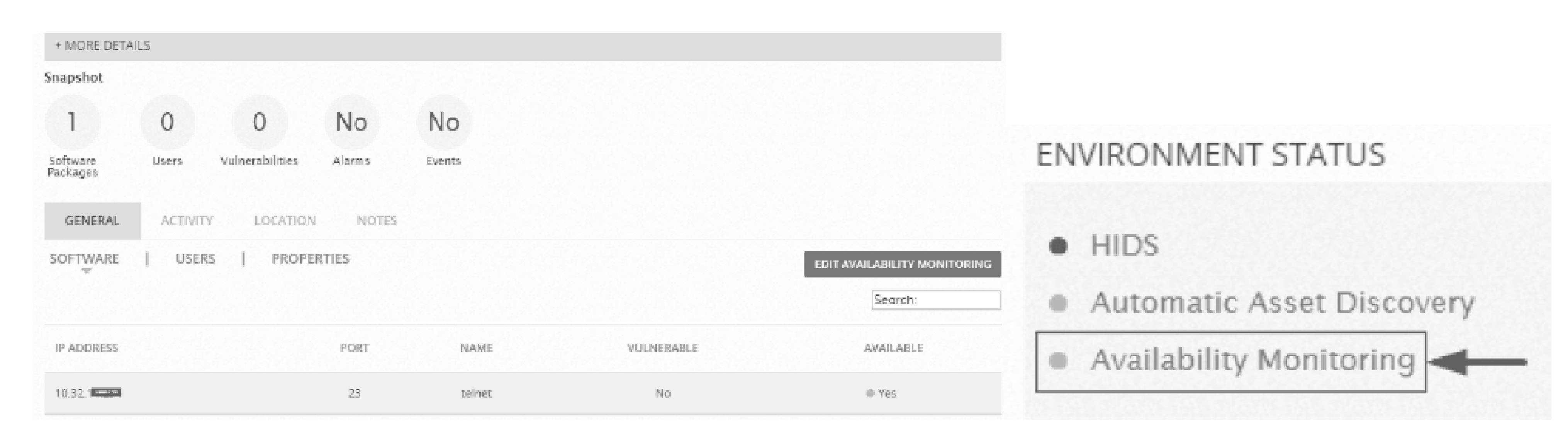


图 8-80　启用监控状态

待以上操作正确完成之后，在 ASSET→Availability 下就能看到监控服务的详情。

8.6.8 Nagios 报错处理

初次使用 OSSIM 进行 Nagios 设置的读者，会在添加资产并开始监控时出现 Nagios 报错问题，例如用户在资产菜单中添加端口监控操作，如图 8-81 所示。

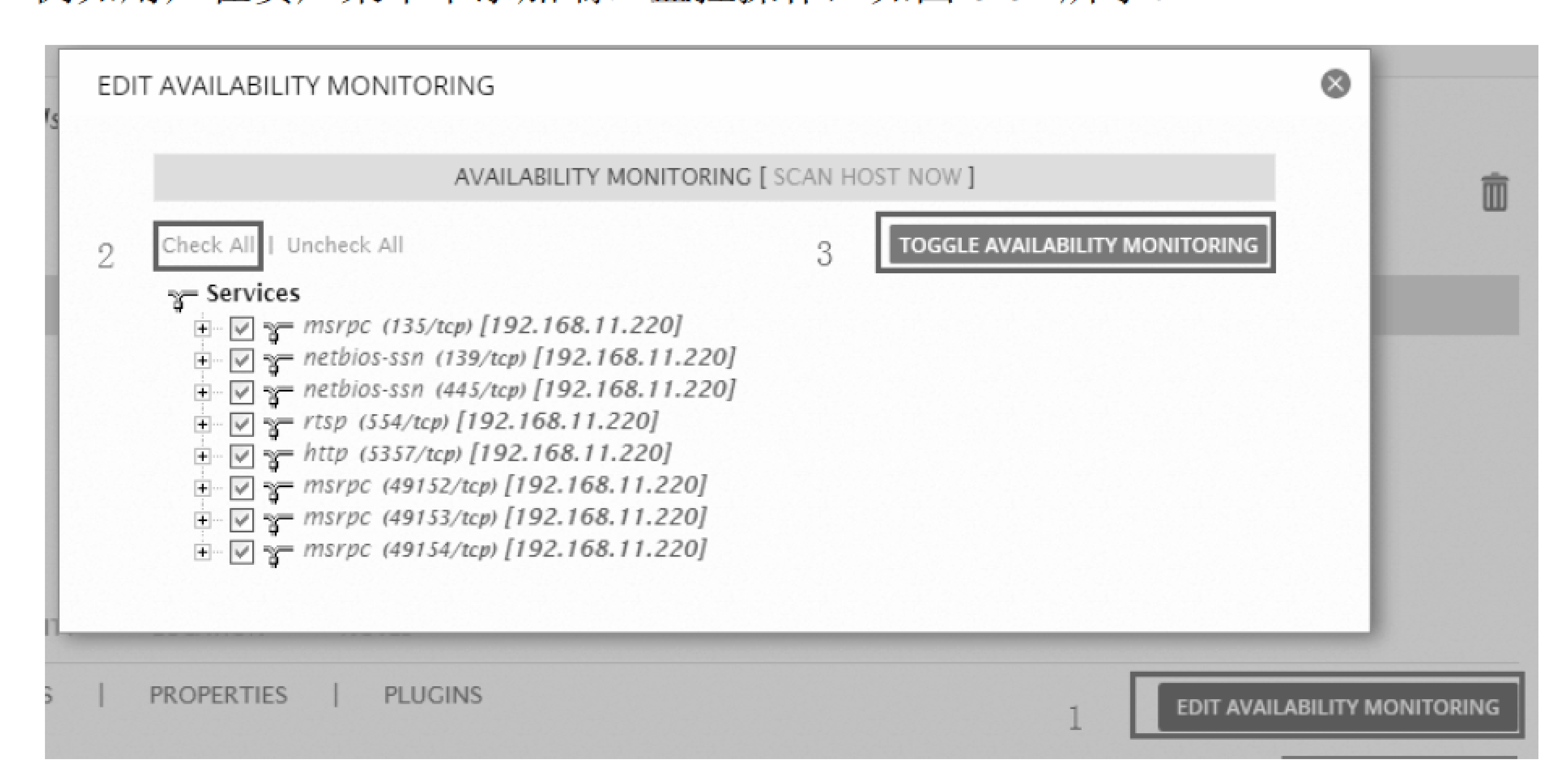


图 8-81　添加端口监控

图 8-80 中所示的三步操作完之后也没有报错，但进入 ENVIRONMENT→AVAILABILITY 就显示报错信息，出现“Error: Could not read host and service status information!”错误如何处理呢？主要看最后一次操作做了什么，在上面的背景中，对 192.168.11.220 这台主机做端口监控的操作，那么肯定出在这台主机的配置文件上，如图 8-82 所示。

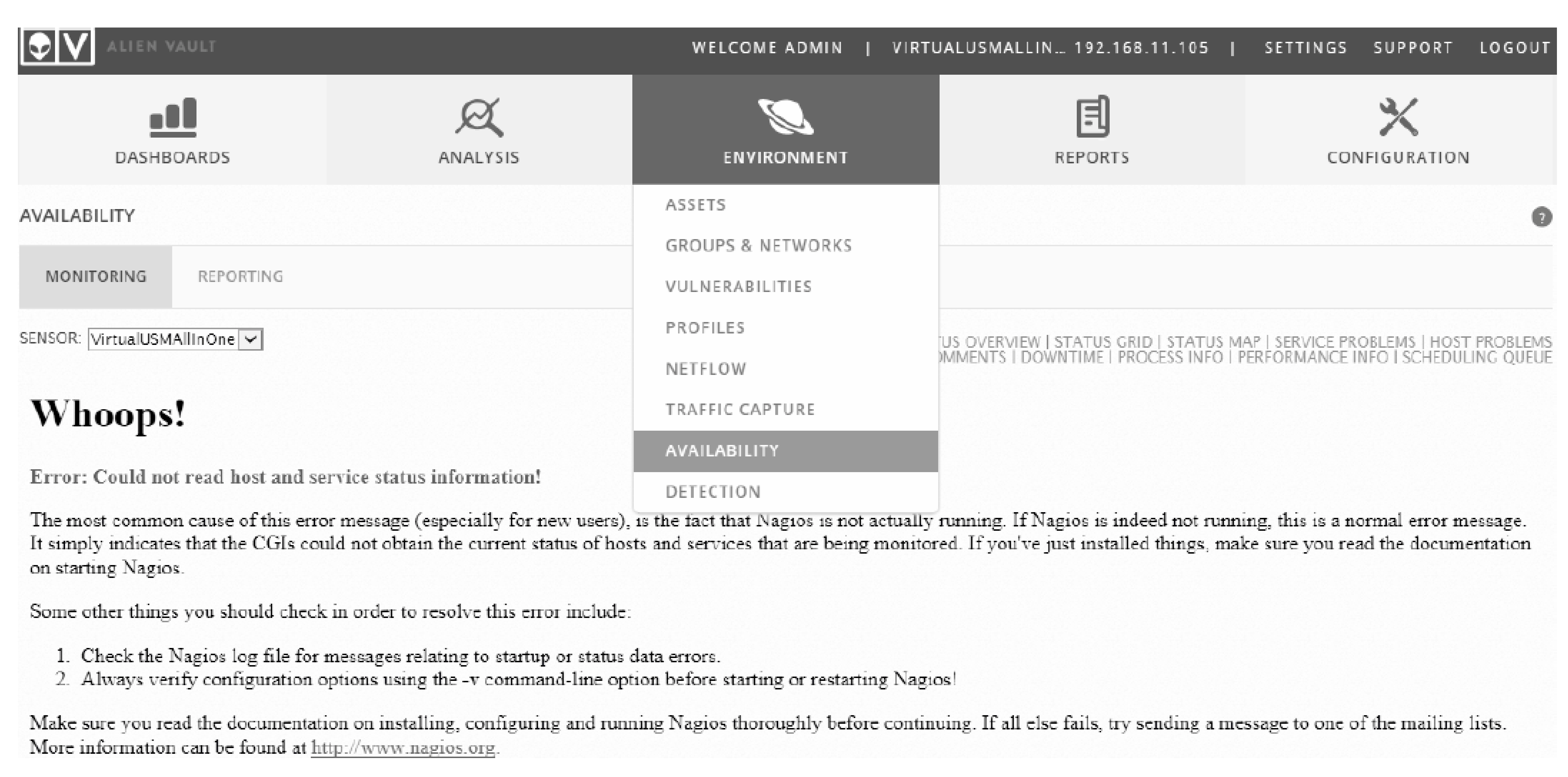


图 8-82　Nagios 故障

下面我们到 OSSIM Server 控制台上，尝试重启 Nagios 服务，观察会有什么情况发生：

```
Processing object config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.105.cfg'...
Processing object config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.1.cfg'...
Processing object config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.89.cfg'...
Processing object config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.40.cfg'...
Processing object config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.220.cfg'...
Warning: Duplicate definition found for host 'localhost' (config file '/etc/nagios3/conf.d/ossim-configs/hosts/192.1
68.11.220.cfg', starting on line 1)
Error: Could not add object property in file '/etc/nagios3/conf.d/ossim-configs/hosts/192.168.11.220.cfg' on line 2.
   Error processing object config files!


***> One or more problems was encountered while processing the config files...

     Check your configuration file(s) to ensure that they contain valid
     directives and data defintions.  If you are upgrading from a previous
     version of Nagios, you should be aware that some variables/definitions
     may have been removed or modified in this version.  Make sure to read
     the HTML documentation regarding the config files, as well as the
     'Whats New' section to find out what has changed.

errors in config! ... failed!
 failed!
VirtualUSMAllInOne:/usr/share/ossim/www/home#
```

这时，我们可以确定问题出在 192.168.11.220.cfg 配置文件上，处理方法是直接修改该文件，或者将其删除重新配置，并且在 Asset 中删除该主机的记录。这样在 Nagios 中遇到的故障即可消除，如图 8-83 所示。

图 8-83　删除主机记录

经过单击上面的删除按钮，系统在后台会删除/etc/nagios3/conf.d/ossim-configs/下的 hosts 及 host-services 配置文件。注意：每次变更时，立即观察结果，切勿多次修改积攒到一起，再进行观察。

8.6.9　被动资产检测 PRADS

OSSIM 下检测资产方式有主动检测和被动检测，主动检测是利用 Nmap 扫描进行扫描，被动检测就包含 p0f、PADS（PADS，Passive Asset Detection System http://passive.sourceforge.net）、prads 等检测方式，这里主要采用了 p0f+PRADS 服务检查资产情况。Prads（Passive Real-time Asset Detection System）是被动实时资产检测系统的简称，它采用数字指纹识别服务，可用于网络资产和实时监测的变化。

Prads 程序可以从每个数据包中获得有用信息，通过查找资产的 MAC、TCP、UPD 以及操作系统指纹和应用相匹配，获得统一状态表（匹配文件参见/etc/prads/tcp-rst.fp tcp-fin.pf 等），并输出各种日志文件。在命令行下输入 prads，然后开始嗅探，若想终止时可以输入^C 命令。

注意："*.pf"文件为指纹库，例如 p0f.pf。

8.6.10 性能监控利器 Munin

监控 Linux 主机的性能需要大量的数据来建立相互关系而得出结论。下面给大家介绍 OSSIM 4.8 中另一个集成工具 Munin，它可以通过客户端－服务器架构，收集数据并将其图形化。Munin 允许跟踪主机的运行记录，就是所谓的"节点"，然后将它们发送到中心服务器，之后你就能以图像形式展示它们。如图 8-84 所示，可以看到一个用 Munin 图形界面显示磁盘 I/O 信息的示例。

图 8-84　显示磁盘 IO

Munin 安装在/etc/munin 目录下，我们先了解一下它的配置文件 munin.conf。

在 OSSIM 4 USM 中打开/etc/munin/munin.conf 文件，其中 dbdir 设置决定 munin 把收集到的 RRD 格式统计数据放在/var/lib/munin 目录下。

htmldir 设置控制 munin 输出数据的位置，在/etc/apache2/conf.d/munin 配置文件中定义了虚拟目录 munin，也就是说我们通过 https://ip/munin 就能用图形化显示监控节点状态。

Logdir 和 rundir 设置控制 munin 的日志文件和 PID 文件放置位置。

最后，我们必须在 munin.conf 文件中定义所有将向服务器发送报告的节点主机，命令为：

```
[hostname.example.com]  address 10.0.0.1  use_node_name yes
[hostname2.example.com]  address 10.0.0.2  use_node_name yes
```

括号中是每个节点的名字，后面紧跟它的 IP 地址，use_node_name 命令控制 munin 命名

节点的方式，如果后面跟的参数为“yes”，表示用括号中的值来命名，如果是“no”则将执行一个 DNS 查询。每个 munin 节点都采用 TCP 4949 端口与 munin 服务器通信，所以你必须确保这个端口在主机防火墙上允许通过。

与 Nagios 类似，Munin 也是采用插件的架构方式来定义监控内容。例如，有专门监控 CPU 的插件，还有监控负载、内存和其他内容的各种插件。Munin 的所有插件清单可以在/etc/munin/plugins 目录下找到，它们以链接到插件的sym-links形式显示。添加一个插件到munin 就是将插件文件的链接写入到/etc/munin/plugins 目录下。我们通过运行 munin-node init 脚本来启动 munin 服务器和节点。

```
#/etc/init.d/munin-node start
#/etc/init.d/munin-node stop
```

这样就能开始 munin 监控并收集需要的数据，munin 周期性地查看每个节点的数据，然后将其上传到 munin 服务器，接着用户通过 munin 的 Web 服务来查看结果。OSSIM 系统中使用 munin 命令如下：

```
#perl /usr/share/munin/munin-graph --cron
```

8.7 Nagios 配置文件

如果打算手动安装 Nagios，首先需要解决 Nagios 的依赖关系问题，尽管它只有很少的依赖关系，而且一部分在 Linux 系统中已安装，Web 前端要求配有支持 CGI 的 Web 服务器，比如 Apache。如果需要让 Nagios 显示图片，则需要安装 libpng、libjpeg 以及 GD 库。但在 OSSIM 系统里，这一切不需要用户完成，只需要将 OSSIM 安装好之后，这一切将自动完成，但对于关键文件位置需要了解，如表 8-10 所示。

表 8-10　Ossim 中 Nagios 文件位置

文件种类	路径
配置文件	/etc/nagios3/nagios.cfg
HTML	/usr/share/nagios3/htdocs/
插件	/usr/lib/nagios/plugins/
守护进程	/usr/sbin/nagios3
CGI 程序	/usr/lib/cgi-bin/nagios3/
锁文件	/var/run/nagios3/
日志文件	/var/log/nagios3/
日志归档	/var/log/nagios3/archives/

当你希望在 OSSIM 中调整 Nagios 时，必须了解 Nagios 配置，Nagios 会在指定时间段调用插件对主机进行监测，对于主机对象的定义、服务的定义、命令及联系人的配置都是必须的，

下面我们加以说明。

Nagios 监控系统中主配置文件是 nagios.cfg，其次就是对象配置文件。Nagios.cfg 会被所有配置文件引用，主要包含影响 Nagios 守护程序运行的一些指令（包括全局配置、日志路径、写入方式、对象配置文件名称等），在对象配置文件中包含了大量对象定义。注意：读者不要随意修改 nagios.cfg 绝对路径以及更改 nagios.cfg 文件名称，如表 8-11 所示。

表 8-11　Nagios 对象定义简介

对象种类	对应配置文件	作用
command	checkcommands.cfg	命令定义与外部程序映射
timeperiod	timeperiods.cfg	时间段定义
host	hosts.cfg	主机定义
hostgroup	hostgroups.cfg	主机组定义（支持中文命名分组）
service	services.cfg	服务定义
servicegroup	servicegroups.cfg	服务组定义
contact	contacts.cfg	主机管理人员
contactgroup	contactgroups.cfg	主机管理人员组
hostdependency	dependencies.cfg	依赖关系定义

Ossim 平台下 Nagios 主配置文件路径为/etc/nagios3/nagios.cfg，在安装 OSSIM 过程中 Nagios 已经配置完毕，剩下的就需要用户将自己的资源加入其中。如果读者独立安装配置过 Nagios，便会体会到其中的艰辛，但在 OSSIM 中实现同样的监控目的，操作过程却非常简单，我们需要关注的配置文件集中在/etc/nagios3/conf.d/ossim-configs/，该目录下包含 hosts、hostgroups、host-service 及 hostgroup-services 配置文件。

8.7.1　主机定义

下面我们观察某个主机定义的例子，前提是我们已经在 OSSIM 中配置添加完成资源并启用资源监控，我们打开/etc/nagios3/conf.d/ossim-configs/hosts/192.168.91.128.cfg 文件，主机定义文件使用了类似 C 语言函数的通用语法，定义由括号{}所包含的指令组成，如图 8-85 所示。

```
define host{
        host_name Host-192-168-91-128
        alias Host-192-168-91-128
        address 192.168.91.128
        use generic-host
        }
~
```

图 8-85　主机定义

- Host_name：定义主机名。
- Alias：定义主机别名。
- Address：指定 IP 地址，如果你使用主机名，则需要考虑 DNS 系统是否完备，否则请使用 IP 地址。

- Use generic-host：Use 表示引用，也就是将主机 generic-host 的所有属性引用到 linux-server 中来，在 nagios 配置中，很多情况下会用到引用。

8.7.2　服务定义

这里我们打开/etc/nagios3/conf.d/ossim-configs/host-services/Host-192-168-91-128FTP.cfg，定义如图 8-86 所示。

```
define service{
        host_name Host-192-168-91-128
        service_description FTP
        check_command check_ftp
        use generic-service
        notification_interval 120
        }
~
```

图 8-86　服务定义

与主机定义相比，在服务描述中没有别名，而使用的是 service_description 指令，试想一下当 A 服务器和 B 服务器都需要同时使用 FTP 服务时，这一条语句就很有效，服务定义就能直接复制使用。

Check_command 命令用于指定服务并进行检测，它是由插件提供的一个子命令，全部集合位于/usr/lib/nagios/plugins/目录中。

Notification_interval 120 代表在主机出现异常后，故障一直没有解决，Nagios 再次对使用者发出通知的时间，单位是分钟。

这台主机有多少服务需要监控，就会有多少个 cfg 配置文件，每个配置文件格式都相同，所不同的是主机名和检测命令，如图 8-87 所示。

```
alienvault:/etc/nagios3/conf.d/ossim-configs/host-services# ls
Host-192-168-91-128FTP.cfg                Host-192-168-91-128GENERIC_TCP_445.cfg
Host-192-168-91-128GENERIC_TCP_135.cfg    Host-192-168-91-128HTTP.cfg
Host-192-168-91-128GENERIC_TCP_139.cfg    Host-192-168-91-128SMTP.cfg
Host-192-168-91-128GENERIC_TCP_443.cfg    Host-192-168-91-2GENERIC_TCP_53.cfg
alienvault:/etc/nagios3/conf.d/ossim-configs/host-services#
```

图 8-87　主机各服务定义文件

注意：当调整 Nagios 配置之后，如何确定正确与否？可以输入以下命令进行检验：

```
#nagios3  -v /etc/nagios3/nagios.cfg
```

该脚本会自动生成检测报告。

8.8　第三方监控工具集成

多数企业已有一套或多套监控系统，已经形成了一套固有的监控体系，这时候如何跟 OSSIM 系统融合在一起呢？这种情况我们需要在 OSSIM 的菜单做文章，对于不同版本的系统

方法不同。注意：需要读者具有一定 PHP 编程基础才能完成本节实验。

8.8.1 OSSIM 2.3 的集成

下面以 OSSIM 2.3 系统为例，讲解怎样将 Cacti 集成到 OSSIM 中，这时我们需要修改 PHP 代码，首先需要安装 Cacti 并配置好，然后我们需要编辑/usr/share/ossim/www/menu_options.php 文件，大约在 1044 行的位置加入如下代码，如图 8-88 所示。系统在设计时，为了便于扩展菜单采用了 PHP 数组菜单，具体如下：

```
$menu["Monitors"][] = array(
    "name" => gettext("Cacti"),
    "id" => "Cacti",
    "url" => "http://192.168.150.100/cacti",
);
$menu["Monitors"][] = array(
    "name" => gettext("Zabbix"),
    "id" => "Zabbix",
    "url" => "http://192.168.150.100/zabbix",
);
```

```
1030            "target" => "main",
1031            "url" => "nagios/index.php?sensor=" . $sensor_nagios["host"],
1032            "help" => "javascript:top.topmenu.new_wind('http://ossim.net/dokuwiki/doku.php?id=user_
     manual:monitors:availability','Help');"
1033        );
1034        $hmenu["Availability"][] = array(
1035            "name" => gettext("Reporting") ,
1036            "id" => "Reporting",
1037            "target" => "main",
1038            "url" => "nagios/index.php?opc=reporting&sensor=" . $sensor_nagios["host"],
1039            "help" => "javascript:top.topmenu.new_wind('http://ossim.net/dokuwiki/doku.php?id=user_
     manual:monitors:availability','Help');"
1040        );
1041 }
1042 if (Session::menu_perms("MenuMonitors", "MonitorsAvailability")) { $monitors = 1;
1043        $menu["Monitors"][] = array(
1044            "name" => gettext("Cacti") ,
1045            "id" => "Cacti",
1046            "url" => "http://192.168.150.100/cacti",
1047        );
1048        $menu["Monitors"][] = array(
1049            "name" => gettext("Zabbix") ,
1050            "id" => "Zabbix",
1051            "url" => "http://192.168.150.100/zabbix",
1052        );
1053        $hmenu["Availability"][] = array(
1054            "name" => gettext("Monitoring") ,
1055            "id" => "Availability",
1056            "target" => "main",
1057            "url" => "nagios/index.php?sensor=" . $sensor_nagios["host"],
1058            "help" => "javascript:top.topmenu.new_wind('http://ossim.net/dokuwiki/doku.php?id=user_
     manual:monitors:availability','Help');"
1059        );
1060        $hmenu["Availability"][] = array(
1061            "name" => gettext("Reporting") ,
1062            "id" => "Reporting",
                                                                        1051,47       78%
```

图 8-88　配置菜单

按照以上脚本修改之后，在系统菜单中集成 Cacti 和 Zabbix 的效果如图 8-89 所示。

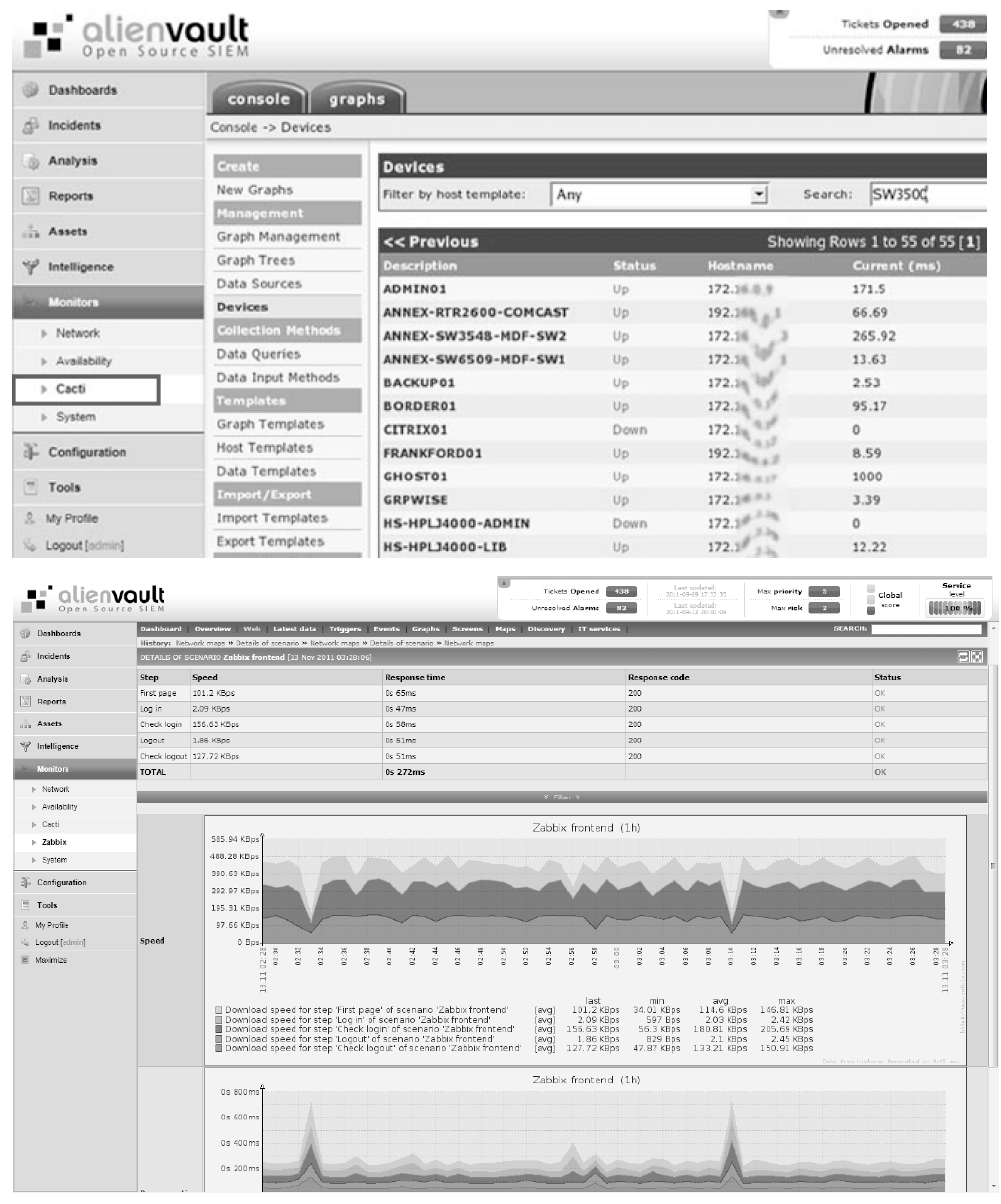


图 8-89　添加菜单成功

8.8.2　OSSIM 4.1 的集成

对于 OSSIM 4.1 系统而言，修改菜单时有些不同，举例来说，修改以下 PHP 代码，首先还是编辑/usr/share/ossim/www/menu_options.php 菜单配置文件，大约在 1316 行的位置加入如图 8-90 所示框中的代码。

效果如图 8-90 右下方所示。注意：本次加入菜单的位置也不是唯一的，这里我们就以加载到 Situational Awareness→Availability 下面为例讲解。修改的方法和 OSSIM 2.3 系统类似，但加入代码和位置都有了一定区别。

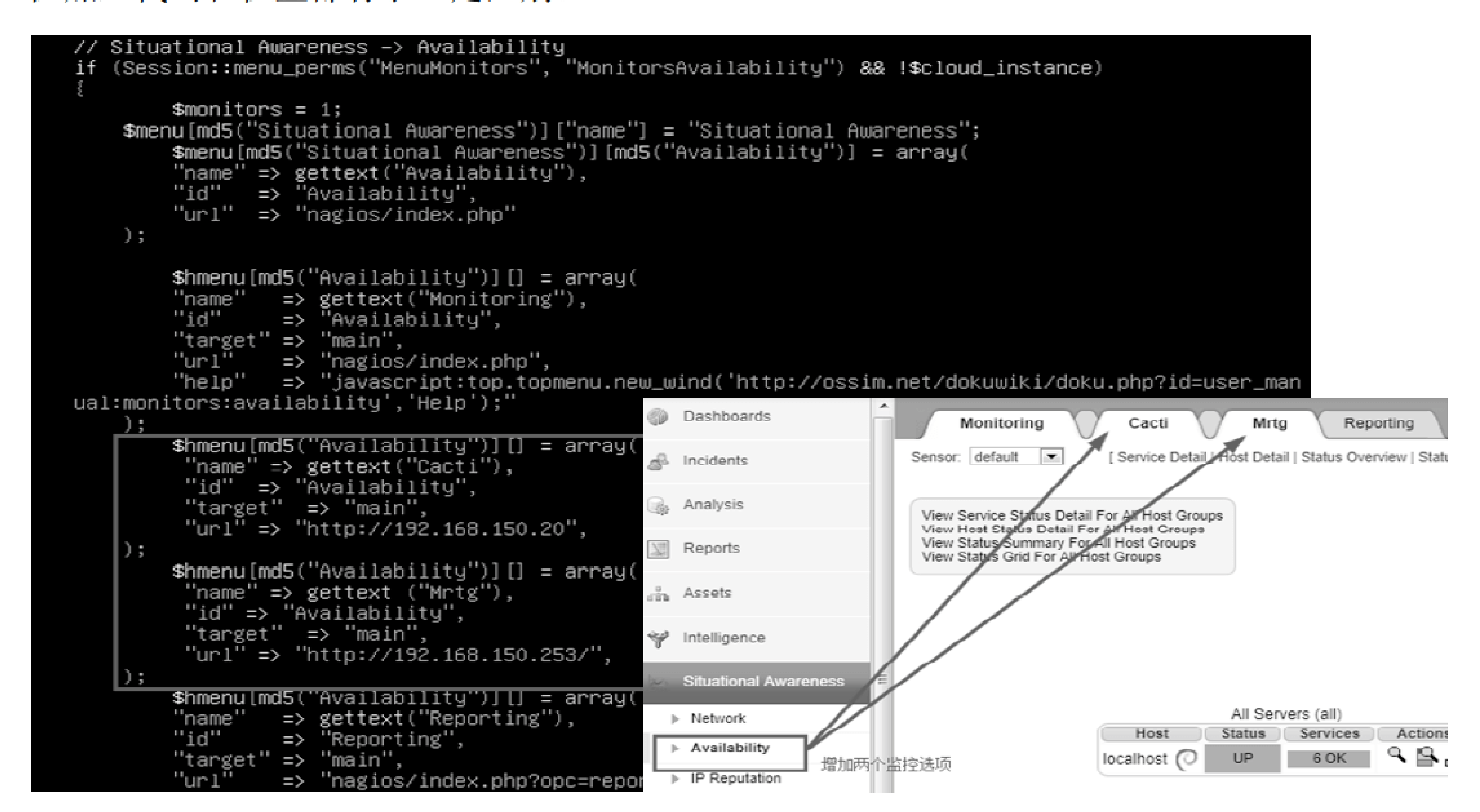


图 8-90　在 OSSIM 4.1 中添加菜单

8.8.3　OSSIM 4.6 的集成

从 OSSIM 4.6 开始，在新的框架中系统菜单调用发生了改变，它的核心文件位于：/usr/share/ossim/include/classes/menu.inc。

下面举个例子，我们在仪表盘菜单加入一行内容。

首先在 Dasboards 添加如下代码，位置在 485 行，如图 8-91 所示。

```
475        /****************************************************************
476        ------------------------- Dashboards ------------------------
477        ****************************************************************/
478
479
480        /* Dashboard permissions */
481
482        /* Overview */
483        $perm_dashboards = $this->check_perm("dashboard-menu", "ControlPanelExecutive");
484
485        /*New menu*/
486        $perm_newmenu= $this->check_perm("dashboard-menu","newmenu");
487
488        /* Deployment status */
489        $perm_deployment_status = $this->is_user_admin();
490        $perm_system_status      = $this->is_user_admin();
```

图 8-91　OSSIM 4.6 中添加菜单

然后在 Deployment status 和 Risk Maps 之间加入如图 8-92 左侧框中的代码，保存后再刷

新页面，会发现在 DASHBOARDS 菜单下多出了一个 NEWMENU 菜单。

图 8-92　OSSIM 4.6 中添加菜单

8.8.4　Sensor 安装 Cacti

由于在 OSSIM 5.1.1 版本中取消了 Ntop 流量监控分析软件，我们在 OSSIM 集成 Cacti 和 Zabbix 流量监控工具，那么在分布式环境中，将其装在 OSSIM Server 端还是 Sensor 端呢？为了减轻 Server 端负载压力，安装在 Sensor 端较为合适，配合 OSSIM 自带的 Nagios，可以更好地对网络流量和应用进行有效监控。在这两节中分别为大家介绍如何在 Sensor 端安装这两款工具。

第 2.3.9 节讲过 OSSIM USM 和 Sensor 安装组件的对比，本节选择在 Sensor 上安装 Cacti，需将 Apache、MySQL 、PHP 组件安装完整。具体操作步骤如下：

（1）安装 Apache

```
#apt-get update
#apt-get install apache2
```

（2）安装 MySQL

```
#apt-get install mysql-server
```

安装过程中会出现输入 root 密码界面，代表输入 MySQL 管理员密码。

（3）安装 PHP5

```
#apt-get install php5
```

（4）安装 php5 组件，让 PHP 支持 MySQL

```
#apt-get install php5-mysql
```

除了 php5-mysql 之外还包括以下组件，用同样方法安装即可：

php5-gd、libjpeg8-dev、php5-imap、php5-ldap、php5-snmp、php5-cgi、php5-adodb、php5-odbc、php*-pear、php*-xml、php5-xmlrpc、php5-mcrypt、php5-mhash、libmcrypt*、libphp-adodb（安装时会提示 include 路径被修改）、libmcrypt-dev、php-fpdf。

另外，将 PHP 配置文件链接到/etc/目录下：

```
#ln -s /etc/php5/apache2/php.ini  /etc/php.ini
```

（5）修改时区

为了解决 date.timezone 的设置问题，根据它的提示来看，只需要找到 php.ini 文件，然后给它赋值即可。如：

```
date.timezone = PRC
```

或者是如下的形式：

```
date.timezone = Asia/Chongqing
#vi /etc/php.ini
Date.timezone="Asia/Shanghai"
#service php-fpm restart
```

重启生效。

（6）测试 SNMP

测试 SNMP 操作如下：

```
#netstat -anup
udp       0     0 0.0.0.0:161        0.0.0.0:*                        7352/snmpd
```

修改配置允许其他主机访问。

Debian 默认只在回环地址上侦听。修改非本地访问则需要修改/etc/default/snmpd 文件，确保以下配置为：

```
SNMPDOPTS='-lSD -lF /dev/null -u snmp -I -smux -p /var/run/snmp.pid'
#snmpwalk -v1  -c public localhost
```

可将配置文件里面 IP 地址修改为被监控的服务器 IP，保存退出。

（7）安装 Cacti

经过以上准备工作之后下面正式安装 Cacti。

```
#apt-get install cacti
```

在 Web Server 类型中选择 Apache2，在出现的“Configure database for cacti with dbconfig-common”提示时选择“Yes”之后，输入你刚才安装 MySQL 数据库的管理员密码，接着提问 cacti 资料库的密码，直接按回车，系统会随机产生一个密码。当然也可以自己指定密码。接着系统会将配置写入到/etc/dbconfig-common/cacti.conf 配置文件中，创建新版配置文件/etc/cacti/debian.php（这就是 Cacti 使用数据库的配置文件）、建库名称为 cacti、创建新版/etc/cacti/apache.conf 配置文件。

查看 Cacti 的版本可以采用命令“apt-cache policy cacti”。

（8）安装 spine

spine 是一个基于 C 语言的，非常快速的轮询引擎。目的是提高 Cacti 的获取数据的性能，操作如下：

```
#apt-get install cacti-spline
```

（9）计划任务

系统在/etc/cron.d/cacti 自动加载了计划任务：

```
*/5 * * * * www-data php /usr/share/cacti/site/poller.php >/dev/null
2>/var/log/cacti/poller-error.log
```

接下来即可通过 Web 访问 http://ip/cacti，按照提示操作即可。安装完成后首次登录，默认账户和密码都为 admin，请自行修改密码和相关用户权限。由于篇幅限制这里就不再详细解释如何在 Web 界面中设置 Cacti，相关内容大家可以到作者博客上查阅。

8.8.5 安装 Zabbix

如果你成功地在 Sensor 中安装完 Cacti，方法类似，步骤如下：

（1）准备工作，首先要确保 8.8.4 节中第 1~6 步正确设置，接着按以下命令操作：

```
#apt-get install fping  libiksemel3
#apt-get install  zabbix-server-mysql
#apt-get install  zabbix-frontend-php
```

根据提示，选择 MySQL 数据库、输入管理密码。在安装过程中，由于 zabbix 需要安装和配置数据库，因此会要求你使用 dbconfig-common 来为 zabbix-server-mysql 配置数据库，回答是。然后输入数据库管理员（DBA）的密码，再输入 zabbix-server-mysql 所用数据库的密码。如果留空的话，则是一个随机生成的密码，重复输入一次后成功安装 Zabbix。

（2）修改配置

修改 php.ini 配置参数：

```
#vi  /etc/php.ini
```

```
post_max_size =16M
max_execution_time =300
max_input_time =300
```

（3）Zabbix 配置

在浏览器上输入 http://ip/zabbix/即可打开 Web 界面，注意首次登录系统用户名采用 admin，密码为 zabbix。接下来需要安装 Agent，Zabbix 需要在被监控的主机上安装 Agent，在 Zabbix 官网上下载相应平台的 Agent 包到各个被监控端。由于篇幅限制，Web 设置和 Agent 的安装不再详细叙述，具体实现大家可以到作者的 51CTO 博客继续学习。

8.9 硬件监控

掌握服务器 CPU、内存、磁盘温度对于系统运维人员非常重要，在 Windows 下有 Everest、鲁大师等软件可以显示机器内部的温度，在一些 HP、IBM 等服务器中也会带一些工具以查询硬件设备的温度。下面教大家几款开源工具用于收集硬件温度。注意：本实验在虚拟机中进行，因找不到传感器所以无效。

8.9.1 IPMI

OSSIM 4.4 以后的版本提供了 IPMI 功能，IPMI 是智能型平台管理接口（Intelligent Platform Management Interface）的缩写，是管理基于 Intel 结构的企业系统中所使用的外围设备采用的一种工业标准。用户可以利用 IPMI 监视服务器的物理健康特征，如温度、电压、风扇工作状态、电源状态等。目前 IBM、HP 及 DELL 等大厂家主流服务器都支持它。它工作时需要和一个专用芯片结合在一起，叫做 BMC 芯片，它最大特点是不依赖于服务器的处理器、BIOS 或操作系统工作。它是一个单独在系统内运行的、无代理管理子系统，只要有 BMC 与 IPMI 固件就可以开始工作。

如果读者在虚拟机上或其他兼容机上安装了 OSSIM，那么 IPMI 服务会因缺少硬件而启动失败，但不影响使用。

这样做的好处在于操作系统未加载的情况下仍然可以进行开关机等底层操作，OSSIM 中提供了 IPMI 软件，是为了支持 BMC 芯片，所以带有 BMC 功能的芯片更适合远程机房的管理，这样可收集服务器风扇转速、CPU 温度等硬件信息。

所以要想 IPMI 系统顺利工作，在服务器上要有支持 BMC 芯片的板卡，操作系统有软件（指 ipmitool）支持，以 Dell R710 为例：

（1）启动服务器，使用 Ctrl+e 进入 ipmi server management configuration，如图 8-93 所示。

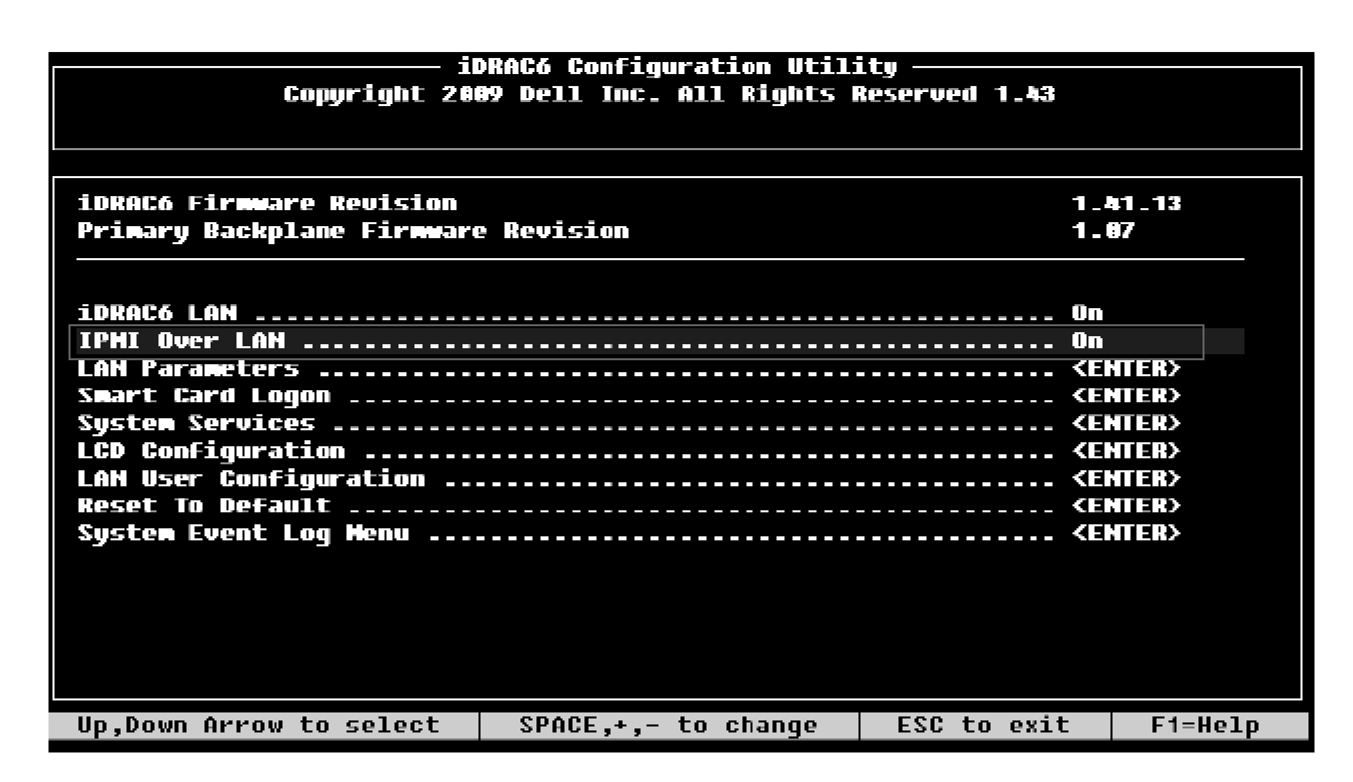


图 8-93　启用 IPMI Over LAN

（2）设置 IPMI Over LAN 为 On。

（3）进入 IPMI Parameters，设置服务器 ip/子网掩码（也可以进去系统通过 Ipmitool 管理软件设置），如图 8-94 所示。

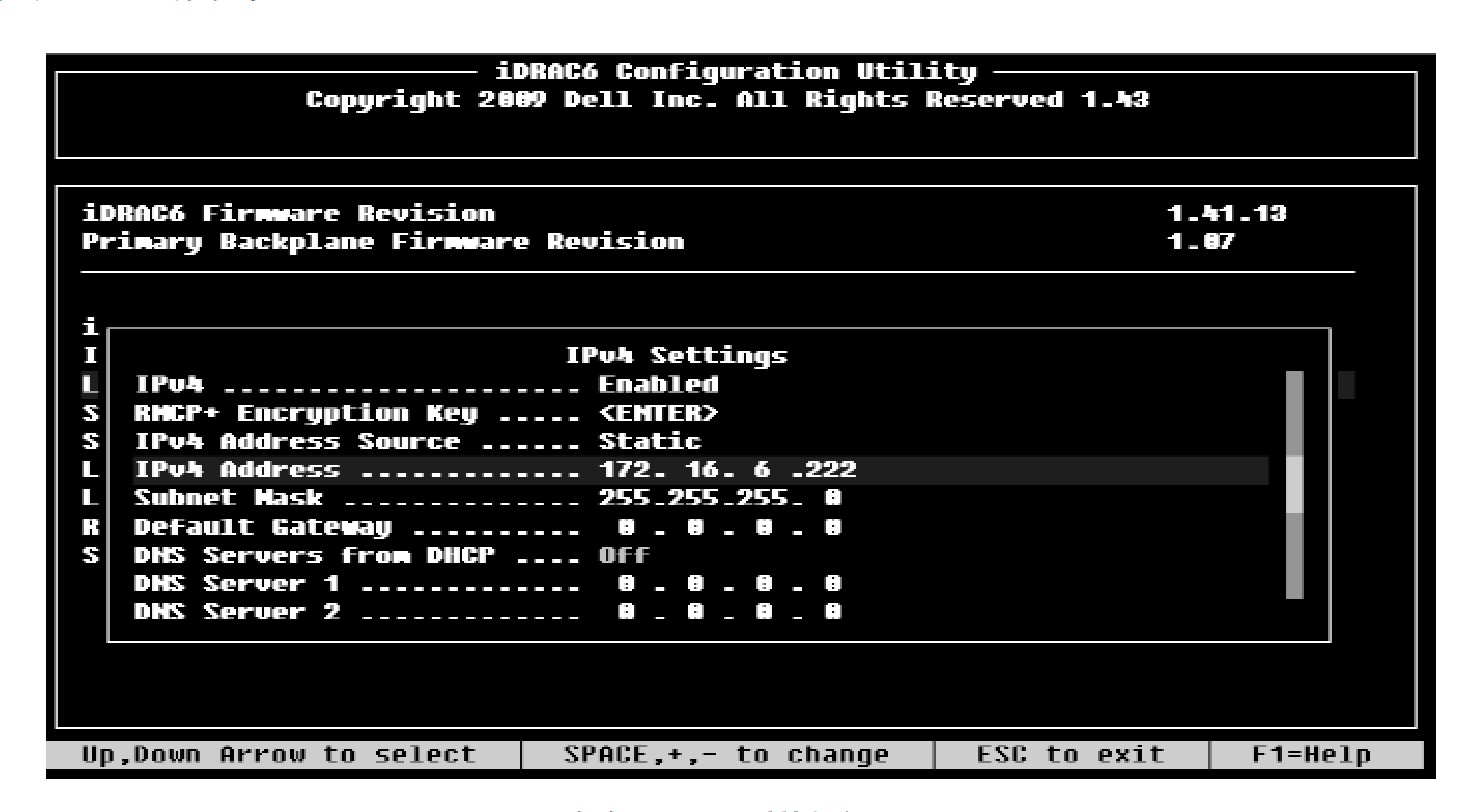


图 8-94　设置 IP

（4）进入 LAN User Confuguration，设置用户名密码（同样也可以进入系统，通过 ipmitool 管理软件设置）。

2. 操作系统提供相应的 IPMI 驱动（OS）

通过操作系统监控服务器自身的 ipmi 信息时，需要系统内核提供相应的支持，OSSIM 系统通过内核对 OpenIPMI（ipmi 驱动）的支持来提供对 ipmi 的系统接口。在使用驱动之前，请先启动该服务：

```
#service ipmi start
```

或者启动模块：

```
modprobe ipmi_msghandler
modprobe ipmi_devintf
modprobe ipmi_si
modprobe ipmi_poweroff
```

```
modprobe ipmi_watchdog
```

所有服务设置完毕后，你就可以在 Web 界面下 Configuration→Deployment→Components→Remote interfaces 进行远程管理。

8.9.2 lm-sensors

lm-sensors 是一款 Linux 的硬件监控软件，可以帮助我们监控主板、CPU 的工作电压，风扇转速、温度等数据。这些数据我们通常在主板的 BIOS 中也可以看到。我们可以在机器运行的时候，通过 lm-sensors 随时监测 CPU 的温度变化，以保护 CPU，防止其过热而烧掉。它的下载地址为 http://www.lm-sensors.org/wiki/Download，目前最新版本 3.3.5。

在 OSSIM 系统中安装方法如下：

```
#apt-get install lm-sensors sensord
#sensors-detect    //查看详细传感器检测结果
#sensors
 adm1023-i2c-0-18
 Adapter: SMBus I801 adapter at 0580
 Board Temp:
 +40° C  (low  =   -55° C, high =  +127° C)
 CPU Temp:    +43° C  (low  =   -55° C, high =  +127° C)
 max1617-i2c-0-4d
 Adapter: SMBus I801 adapter at 0580
 Board Temp:
 +38° C  (low  =   -55° C, high =  +127° C)
 CPU Temp:    +45° C  (low  =   -55° C, high =  +127° C)
```

一般来说，温度在 70 度以下都正常。

如果你的 OSSIM Server 安装了 GNOME 桌面管理器，那么可以通过以下方式获得图形化界面。

```
#apt-get install xsensors sensors-applet
```

8.9.3 hddtemp

通过 hddtemp 可以获取硬盘的温度。安装命令如下：

```
#apt-get install hddtemp
```

比如，服务器磁盘设备名为/dev/sda，要获取它的温度，操作命令如下：

```
#hddtemp /dev/sda
/dev/sda: ST9500325AS: 36°C
```

硬盘必须支持 S.M.A.R.T（自我监测、分析及报告技术）。

8.10 小结

本章详细分析了 NetFlow 采集业务流量的原理和收集方法，列举了 NetFlow 在蠕虫病毒检测的应用，重点介绍了异常流量的分析方法。同时详细介绍了 OSSIM 集成的两款重量级开源工具 Ntop 和 Nagios 的应用和处理故障的方法。除此之外，对 OSSIM 集成第三方监控软件的方法也进行了讲解。

第 9 章

OSSIM应用实战

从本章节可以学习到：

- OSSIM 4.2 与 4.8 版的 Web UI 差异
- OSSIM 站点结构
- SIEM 控制台日志过滤方法
- 仪表盘显示与识别
- SQL 注入攻击行为识别及案例分析
- 资产管理
- OSSEC 安装
- Openvas 漏洞扫描
- 蠕虫异常流量监测
- 报表的输出
- 合规管理 ISO27001 、PCI-DSS 2.0

9.1 使用 OSSIM 系统

用户接触 OSSIM 平台最多的是 Web UI，通过 Web 以可视化方式轻松获取各种安全分析的图表，作为普通运维或监控人员，绝大多数操作都通过 Web UI 来完成，本章详细讲解如何使用这些操作。同时满足一些开发者的需要，某些关键部分的内容会深入讲解。

9.1.1 初识 OSSIM Web UI

如果读者部署过OpenStack基础架构即服务（IaaS）的开源解决方案，可能知道在OpenStack中附带了一个仪表板 Web 应用程序，在这个 Web UI 上执行启动/关闭虚拟机实例，非常方便。OSSIM 提供了更美观的统一交互式图形界面，通过这个界面，管理员可以轻松查看、配置功能模块，包括定义各种资产，查看安全策略、编写关联规则，还可以将你现有的各种 B/S 系统集成进来。随着 OSSIM 版本不断升级，其 Web UI 上显示的各种仪表盘参数位置也会有所调整，但万变不离其宗，读者在理解了本节介绍的内容后，就能自己适应今后版本的变化。

当 OSSIM 系统安装完毕，重启后进入登录界面，上面将显示登录 IP，这时就可以在客户机上登录 OSSIM Web 界面，在浏览器地址栏中输入 https://IP/，首次登录系统输入用户 admin，这时系统提示修改密码。

图 9-1 为 OSSIM 4.2 系统的登录界面，可以明显看出分为 10 个部分，各部分功能如下：

图 9-1　OSSIM 4.2 系统使用界面

（1）导航栏将 OSSIM 的主要功能显示在 10 个模块中，每个模块都可以用鼠标灵活拖放到其他位置以符合个人浏览习惯。实现这些模块的主要代码在/usr/share/ossim/www/目录下，每个子系统对应的目录和文件参见 9.2 节内容。

（2）这部分反映了所监控网络的健康程度，在图中系统监控程度标记为红色，表示威胁严重。如图 9-2 所示。

图 9-2　监控网络健康程度

在 OSSIM 4.8 系统中，通过 Threat Level（威胁等级）显示网络威胁度量情况。这里的 Environment Snapshot 系统环境快照反映出 Web UI 下 Environment 菜单下的总体情况，默认每

隔 5 秒刷新一次数据，其调用源码文件参见/usr/share/ossim/www/home/sidebar_ajax.php。

（3）在图 9-1 中③、④、⑤、⑥、⑦、⑧几个部分的内容，属于整个系统的数据展现层，可使运维人员快速了解各种趋势图、汇总表、TOP N 、安全事件、安全态势等信息。

在图 9-1 中的③部分信息反映出当天的安全事件和日志事件的数量，单击对应区域就能查看详情。用波浪图的方式反映了日志和安全事件增长趋势。这些数据来源于后台数据库 alienvault.alarm，还包括 Dashboards→Overview→Security 下的 Security events:Top Alarms 的内容。

（4）系统对告警进行统计分析并得出目前的状态。在第 4 章讲过 OSSIM 将网络威胁级别从低到高分为 Low、Precaution、Elevated、High、Very High 共 5 个等级（网络威胁详情见 1.1.1）。要让计算机能够识别威胁，一种可行的方法就是将威胁进行等级化，将漏洞进行威胁等级划分，对每个漏洞给出攻击它所需要的等级，以及攻击成功后可以达到的等级，这样就可以通过规则和漏洞库相结合来实现。

在 OSSIM 首页单击仪表盘，将进入 Risk Metrics 风险度量界面。如图 9-3 所示。这个界面上会具体显示设备和网段的风险值。现实的服务等级为 100%表示，所监控网络比较安全，这个值越低表示网络越不可靠。

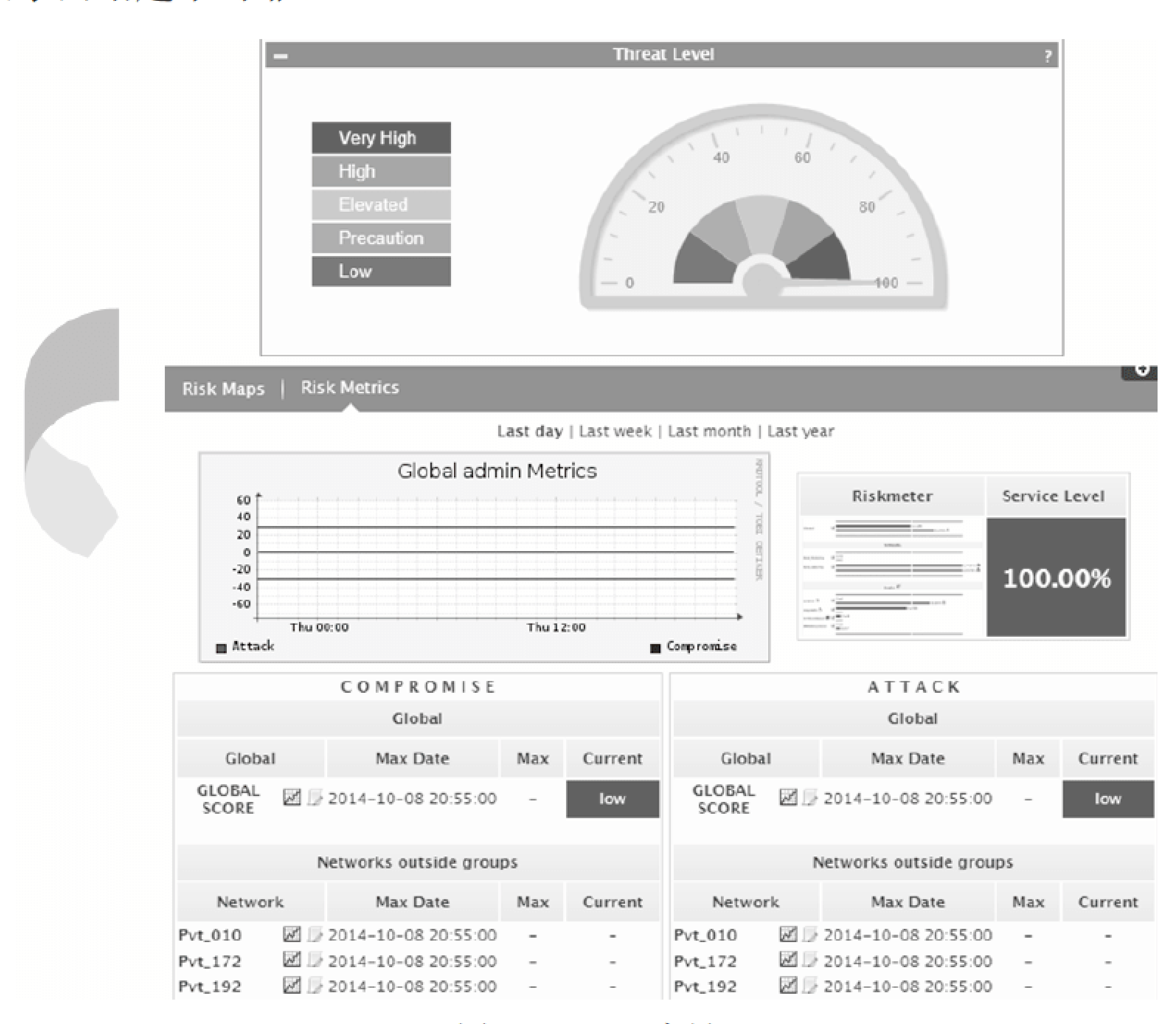


图 9-3　Risk 度量

（5）这部分将分布在网络中的 4 个探针获取的数据和数据源（DS 位于 Deployment→Collection→Data Sources）进行比较，并根据分类发出报警显示在图中主界面上，用不同颜色区分不同的探测器，当用鼠标单击其中的圆点就会显示 SIEM 详细信息。对于 OSSIM 4.8 以上版本，数据源的 Web 路径为 Configuration→Threat Intelligence→Data Source。

（6）这部分显示系统中疑似警报（一些比较可疑的报警信息）和打开的通知单。

（7）这部分对 SIEM 事件信息显示产品类型的 Top10。OSSIM 的产品类型包括：Alarm、Anomaly Detection、Application、Application Firewall、Authentication and DHCP、Data Protection、Database、Endpoint Security、Firewall、Honeypot、Infrastructure Monitoring、Intrusion Detection、Intrusion Prevention、Mail Security、Mail Server、Management Platform、Network Access Control、Network Discovery、Operation System、Other Devices、Proxy、Remote Application Access、Router/Switch、Server、Unified threat management、VPN、Vulnerability Scanner、Web Server 共 28 个大类。

（8）此部分就是用饼图显示事件类别的 Top10。

（9）此按钮显示整个系统数据快照，侧重安全事件，方便掌握工作状况。

（10）此按钮能显示系统配置信息。

在 OSSIM 4.3 系统中虽然 Web UI 界面发生了变化，即把菜单栏由传统放置在左边调整到了屏幕上端，菜单进行了合理地优化。

9.1.2　OSSIM 4.8 界面

读者在参考前面介绍的 OSSIM 4.2 版本后继续使用 OSSIM 4.8 版本，对前端界面，会有更加深入的体会，新版中无论在功能上，还是在界面风格上，比以往版本有了较大改善，其功能设计合理，主要栏目 Dashboards、Analysis、Environment、Reports、Configuration 都体现在浏览器的顶栏。OSSIM 4.8 界面如图 9-4 所示。

图 9-4　OSSIM 4.8 界面

安全事件与日志曲线特点，从正常网络运行获得的监控事件数据和日志数据分析，有如下情况应引起管理员注意，首先这两条波浪线可持续，不能发生突变（无故中断或突然增大或减小），类似图 9-5 所示的情况属于不正常现象。

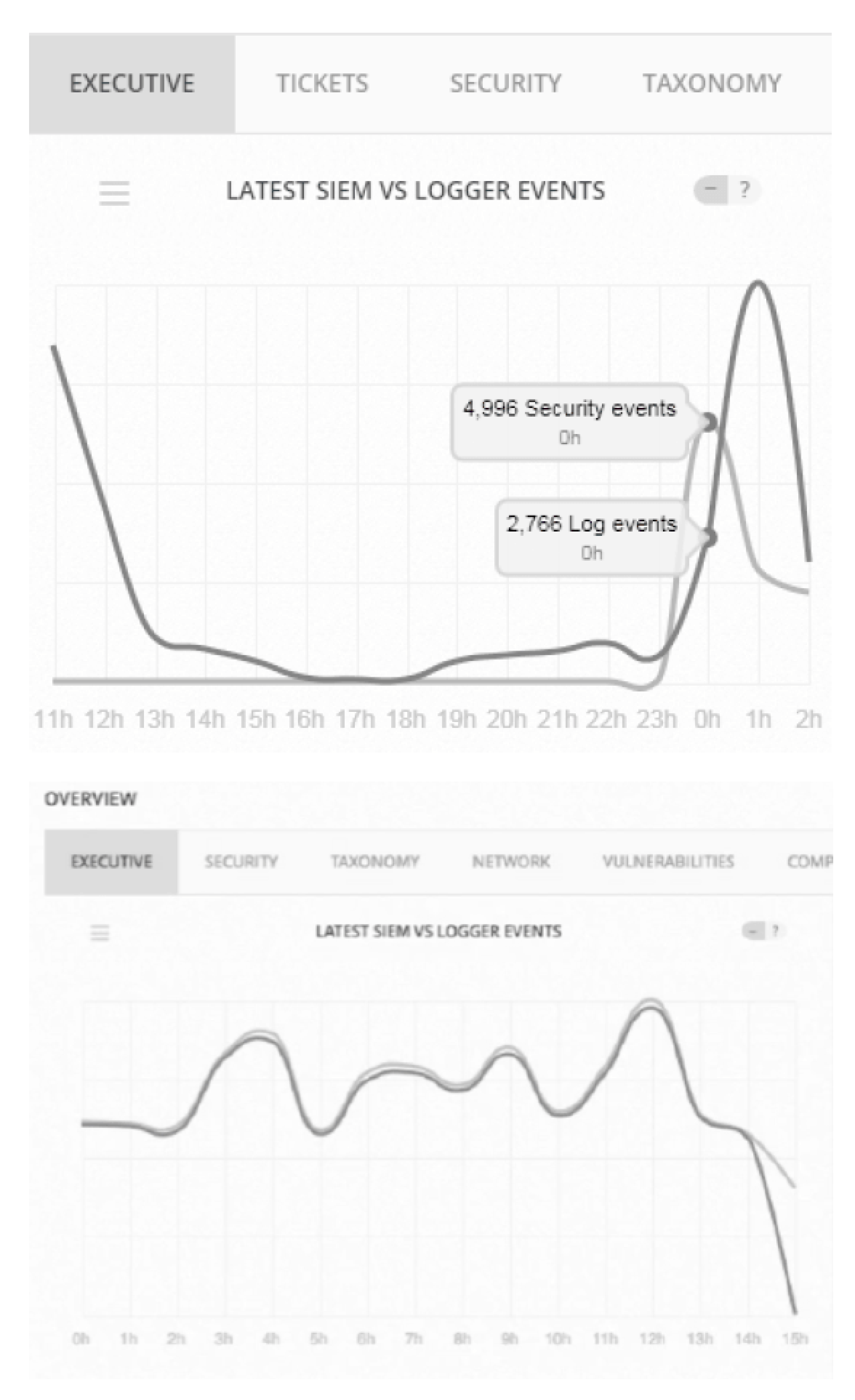


图 9-5　日志曲线发生突变

其次这两条线所反映的数值是相近的，蓝色曲线反映了日志随时间波动，绿色曲线反映了安全事件随时间的波动情况，这两条线就如同鸳鸯是比翼双飞。如果两条线上某一点反映的数据大相径庭（安全事件数量很大，而日志条目非常小），说明可能是某些事件或日志的数据发生中断或被删除。如果某时间段 EPS 突然变化则意味着日志流被阻断。

最后单击曲线上的关键点，可查看事件或日志的具体数值，单击关键点能看到具体的事件信息，如果单击后反应迟钝或长时间无响应，很有可能是后台数据库 I/O 遇到瓶颈。仅凭菜单读者不易对 OSSIM 功能的全貌进行了解，下面采用思维导图的方式将它的主要功能予以展示。如图 9-6 所示。

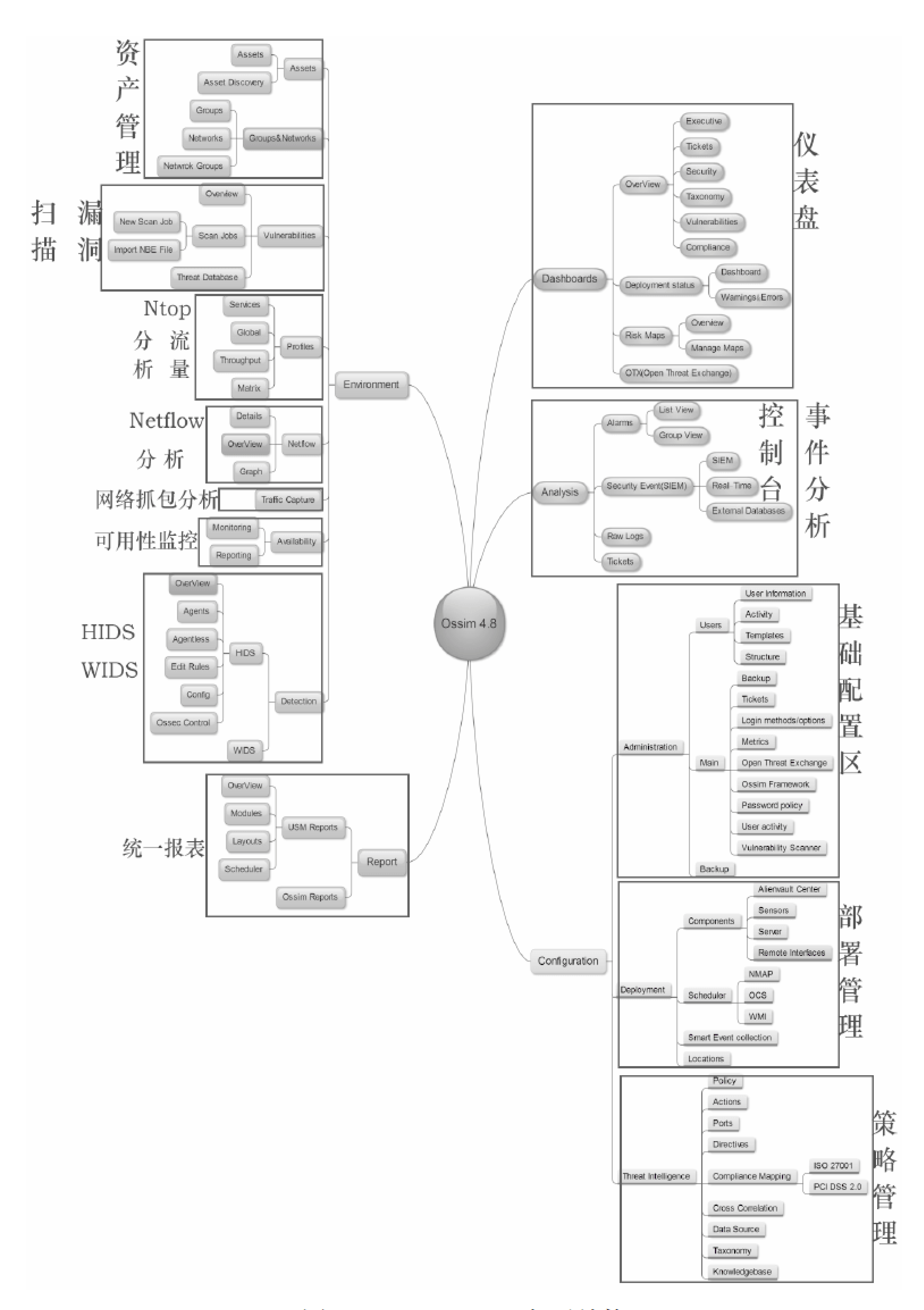


图 9-6　OSSIM 4.8 主要结构

由于篇幅限制有关菜单的描述大家可以访问 http://chenguang.blog.51cto.com/350944/1579815。

9.1.3　OSSIM 控制中心：AlienVault Center

OSSIM 的控制中心菜单主要集中在上面图 9-6 所示的部署管理区域，控制中心主要作用就是向管理员报告当前系统和网络状态，以便管理员了解 Server 和 Sensor 的各项关键参数，具体访问方式在 Configuration→Deployment→Alienvault Center，其访问的 AC（Alienvault Center）信息存储在 OSSIM 库的/var/lib/mysql/avcenter/中，这里主要了解整个系统状态，如图 9-7 所示。

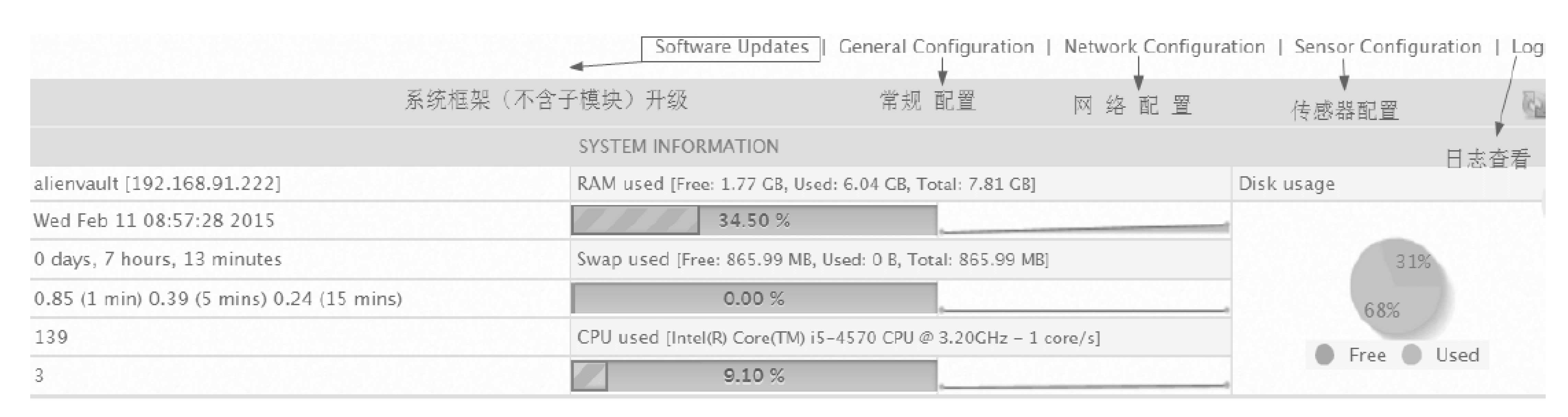


图 9-7　系统各项配置

从图 9-7 中也能了解各个网络接口卡的配置，流量统计信息，软件系统状况，包括 NTP 的配置情况，这里还包含了传感器的插件配置，工作状态以及系统详细日志情况。

9.1.4　基于角色的访问权限控制

OSSIM 系统提供了灵活的多用户角色管理，不同角色具有不同的权限，实现方法为选择 Configuration→Administration→Users，如图 9-8 所示。

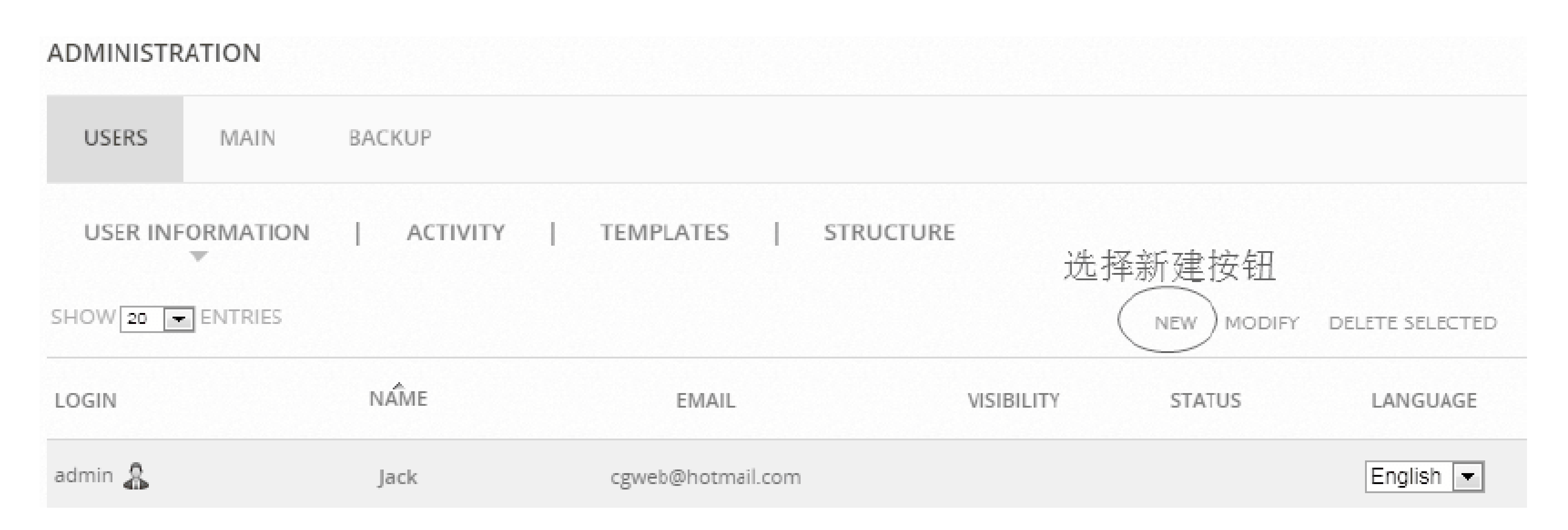


图 9-8　新建用户

首先，选择新建模板，模板名称为“test”，如图 9-9 所示。

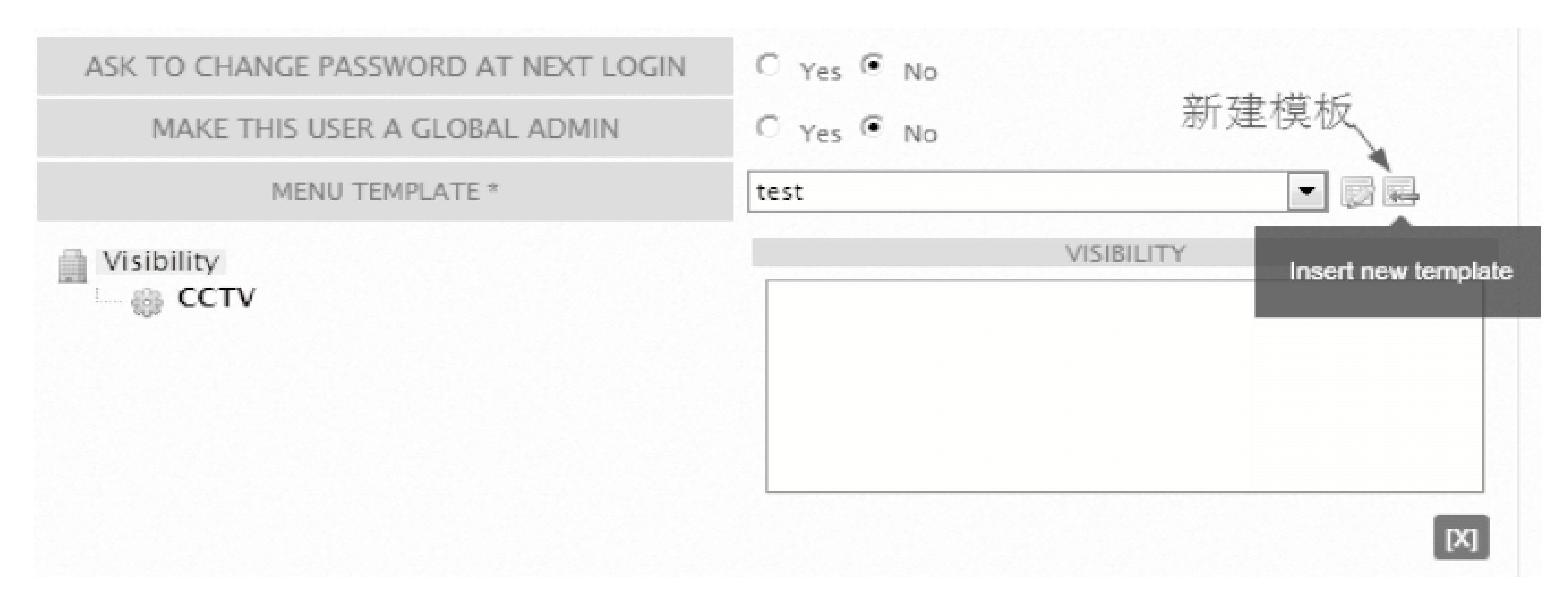


图 9-9　新建模板

接着输入可见性 Visibility，这里就是注册时的组织名称，本实例中为“CCTV”，然后选择组织的资源，可以选择单台主机，也可以是一个网段，最后在列表中选择资源所在网段内的传感器，如图 9-10、图 9-11、图 9-12 所示。

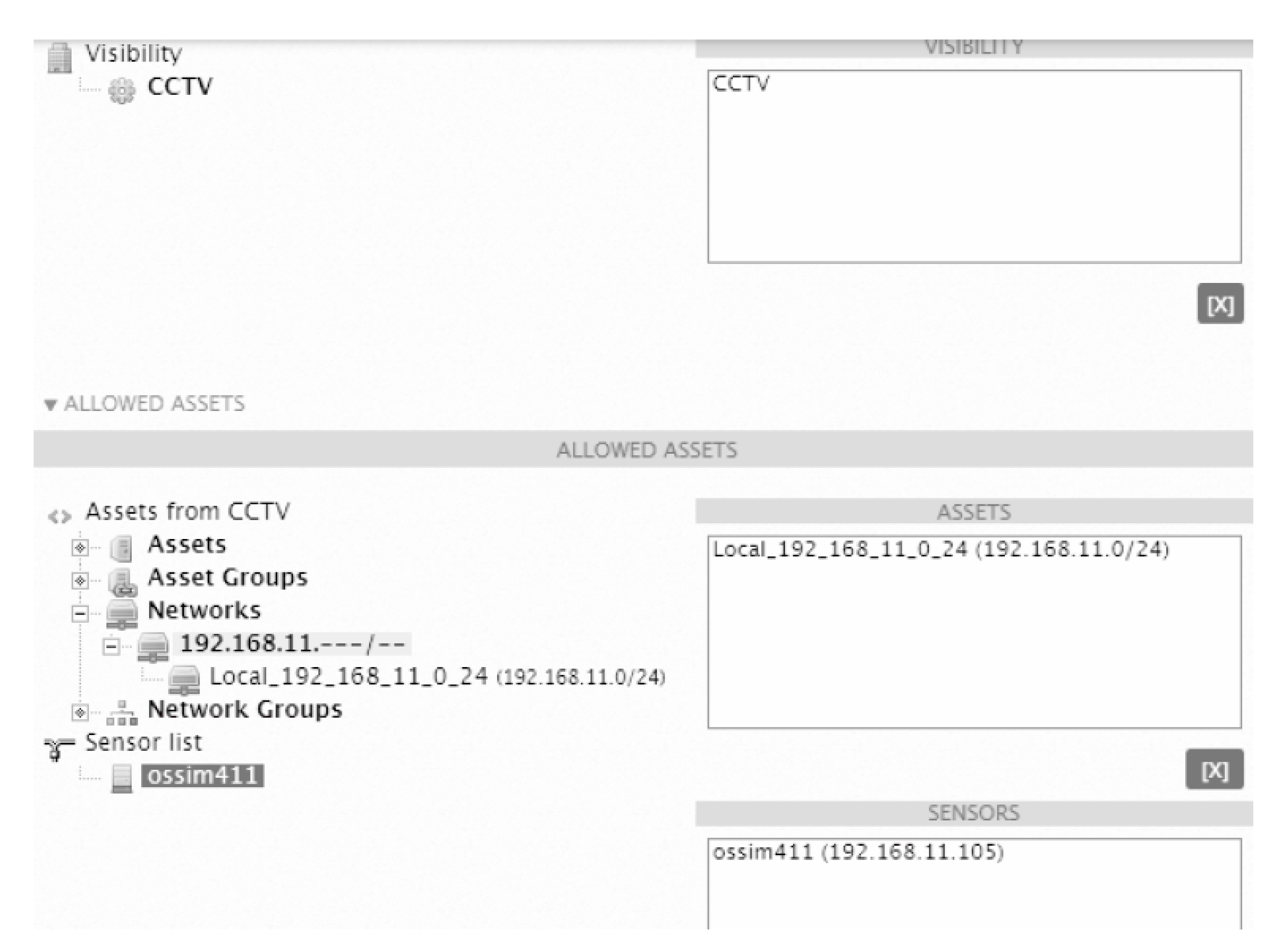


图 9-10　选取可见性 Visibility 与所监控的网段

NEW TEMPLATE

Name: CANCEL SAVE TEMPLATE

SELECT / UNSELECT ALL

- [x] Dashboard -> Overview
- [x] Dashboard -> Overview -> Manage Dashboards
- [x] Dashboard -> Overview -> Metrics
- [ ] Dashboard -> Risk Maps
- [ ] Dashboard -> Risk Maps -> Manage Risk Maps
- [ ] Dashboard -> OTX (Open Threat Exchange)

图 9-11　选择需要的菜单

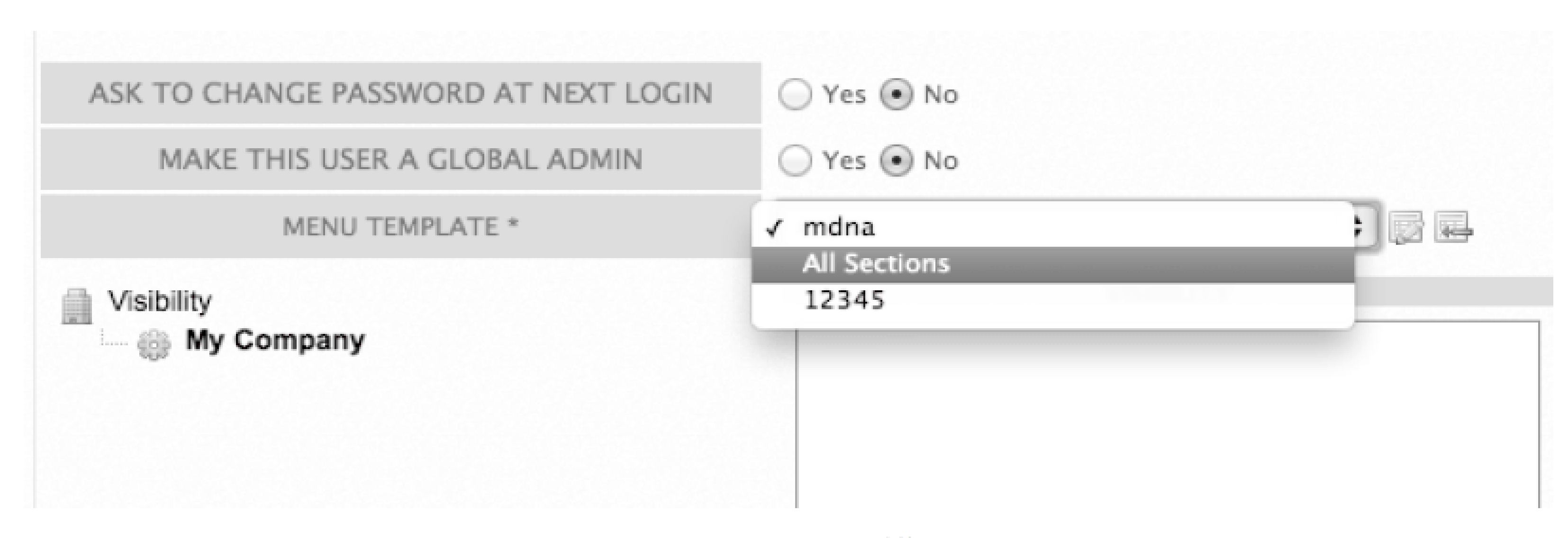


图 9-12　选择模板

当选择模板后，接着选取 Visibility→MyCompany，以及所监控的资产，下一步选定 Sensor，最后输入 admin 用户密码，添加用户工作即完成，如图 9-13 所示。

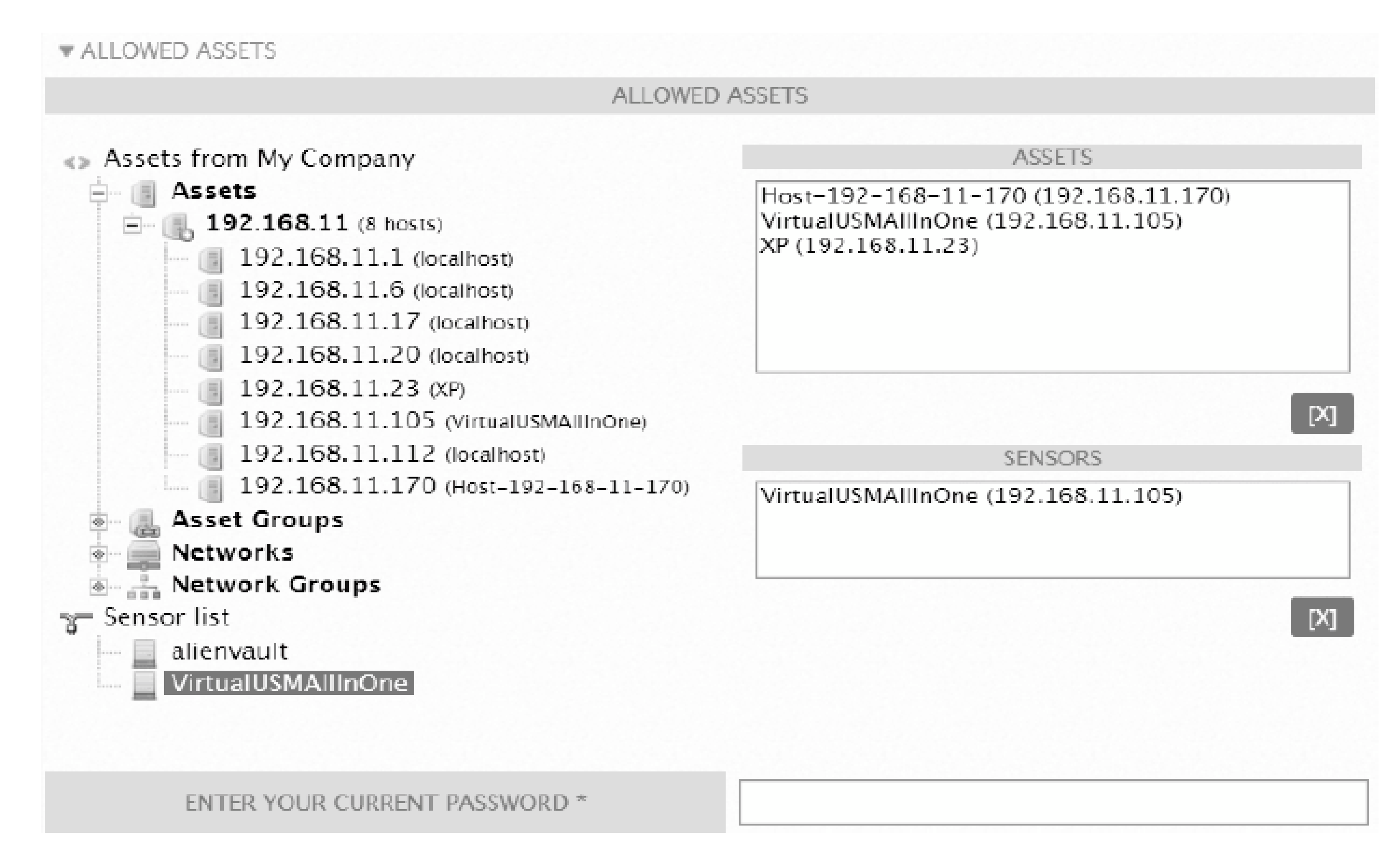


图 9-13　设定资产和 Sensor

用户添加完毕，需要对用户权限进行实际登录测试，因为有些菜单的内容之间相互关联，比如 Analysis→alarm 和 SIEM 之间存在内在联系，所以指派选项时需要全选，效果如图 9-14 所示。在 Dashboards 仪表盘展示的各种可视化内容，也是从各种安全事件中抽取，所以在初次分配权限时，也应当选择，而对于 Environment、Reports、Configuration 设计资产、流量和系统配置管理的部分，则可以分配给权限更高的用户（仅次于 admin）。

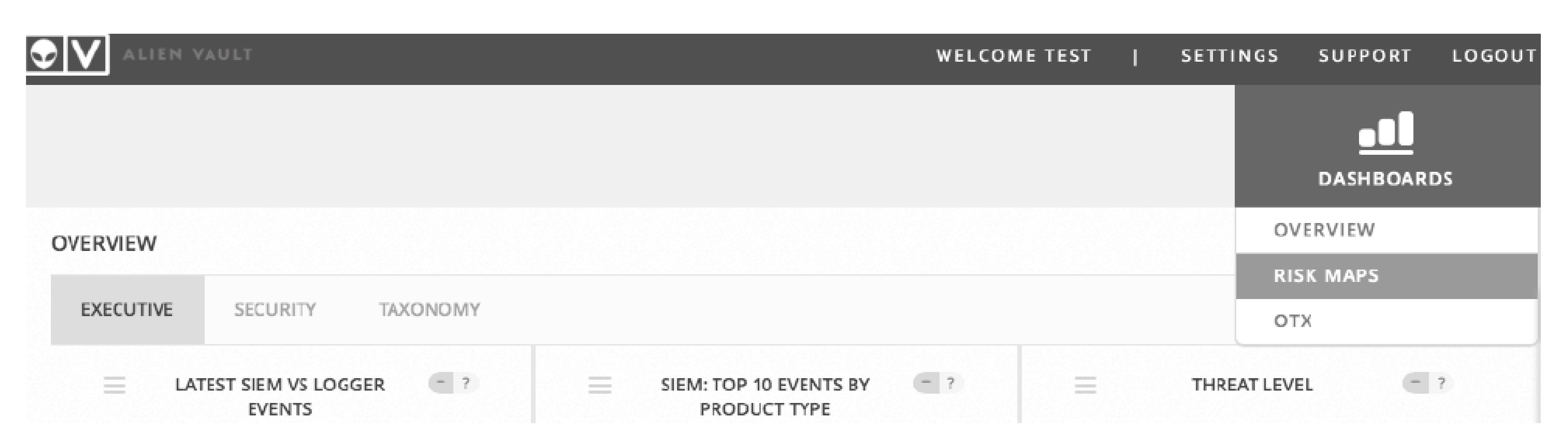


图 9-14　验证效果

通过多用户管理可以将繁杂的安全分析过程，分解到多个用户的控制台上，这样可以加快分析进度和准确性。注意：以上讲述了对于初次涉及权限设定的用户，通常我们可以在 Templates 中实现，然后在新建用户时直接调用模板，这样操作更方便。如图 9-15 所示。

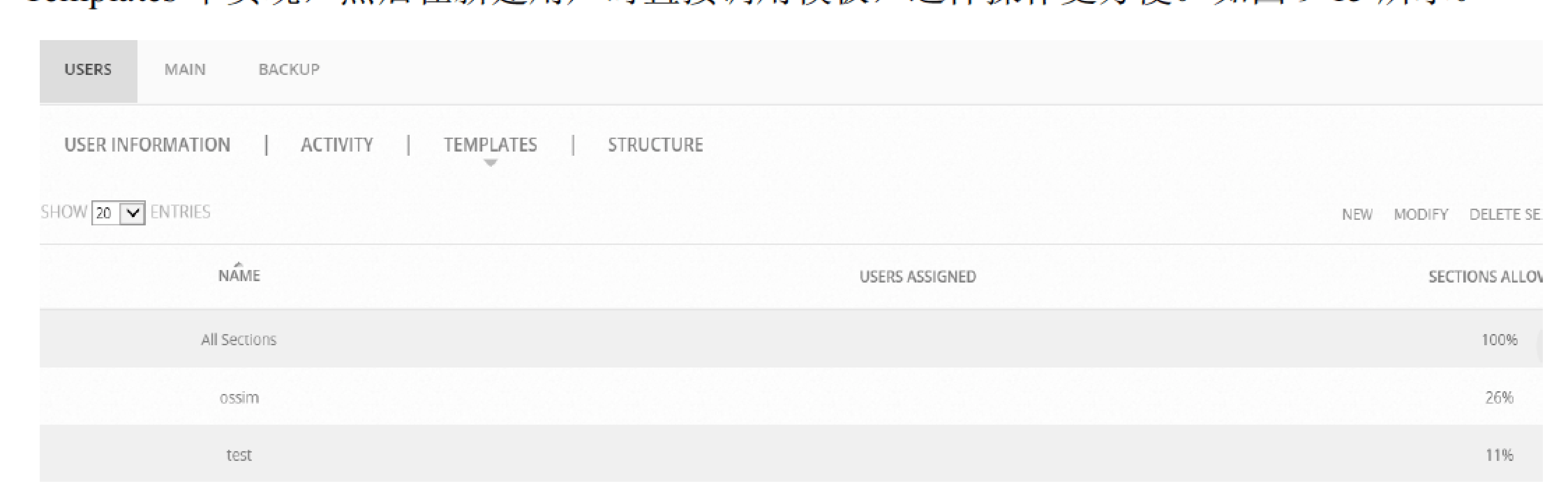


图 9-15　用户权限

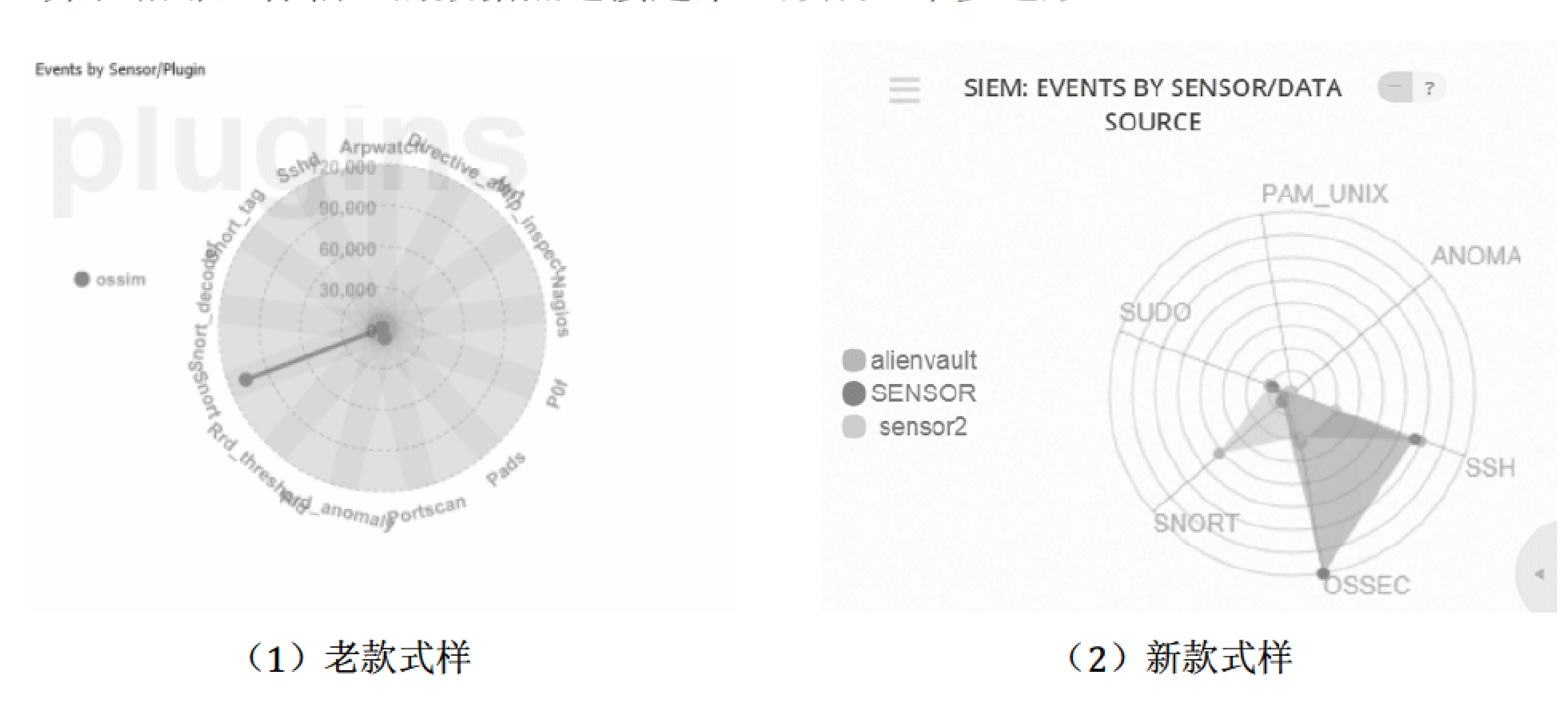
图中明确地标识出每个用户所控制系统的比例，admin 用户拥有完全控制权，OSSIM 用户拥有 26%的控制权，单击某个用户还可查看详情。

9.1.5　仪表盘详解

OSSIM 系统可以通过仪表盘展示数据视图，而且可以自定义。仪表盘（Dashboard）是一类模仿汽车速度表的一系列图表，特点是简单、直观和高效。下面看看 OSSIM 几个典型图例：

（1）雷达图

雷达图也称为蜘蛛图或星状图，如图 9-16 所示，它是一种以二维形式展示多维数据的图形。雷达图从中心点辐射出多条坐标轴，每一份多维数据，在每一维度上的数值都占用一条坐标轴，并和相邻坐标轴上的数据点连接起来，形成一个多边形。



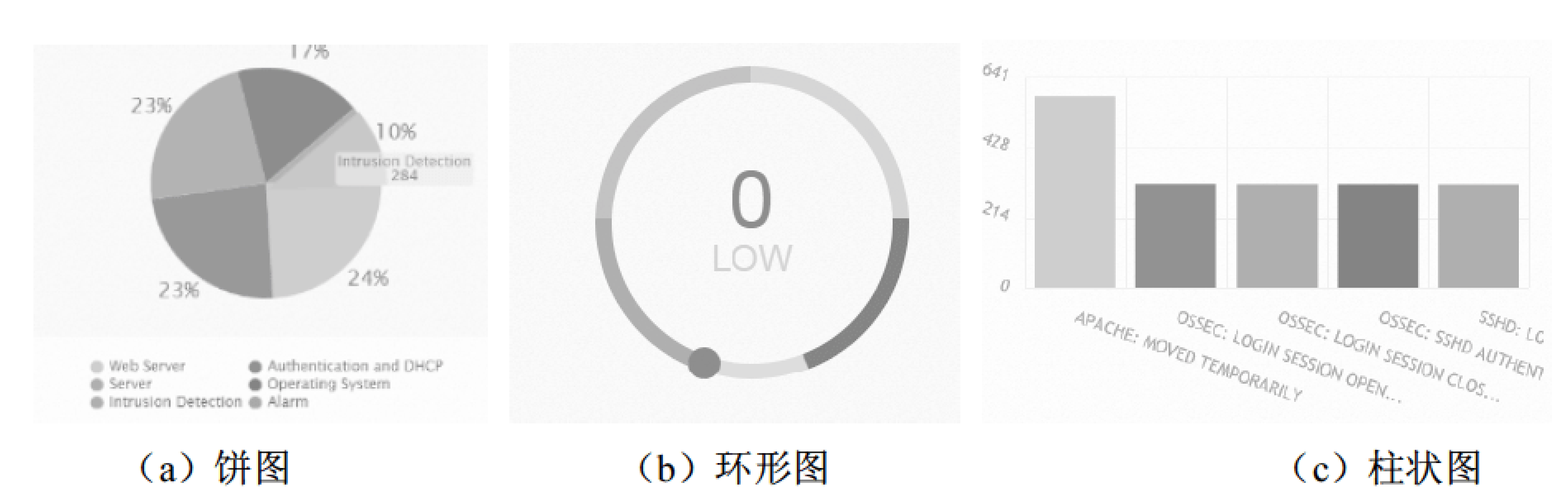
（1）老款式样　　　　（2）新款式样

图 9-16　新旧雷达图对比

雷达图周边是按插件统计的事件名称、点、线、面所连成的区域可以分析出事件的多少，从而看出严重性，如果将相邻坐标轴上的刻度点也连接起来以便于读取数值。雷达图的特点，数量越大越靠近边缘，越小越接近圆心。在 OSSIM 中，不同传感器收集到的事件通过不同颜色可以反映在一个雷达图内。

（2）饼图、环形图、柱状图

数据可视化能帮助用户理解数据，在 OSSIM 仪表盘中大量使用了各种表现形式的图表，其中又以饼图、环形图、柱状图和波浪图最为常用，如图 9-17 所示。



（a）饼图　　（b）环形图　　（c）柱状图

（d）条状图　　（e）气泡图　　（f）波浪图

图 9-17　OSSIM Web UI 中的常用图形

在 OSSIM 的仪表盘上，采用了饼图和柱状图形式，因为人们在理解这种形象的图形比字符要容易得多。大家或许会问，仪表盘中为什么有饼图和柱状图等多种表现形式呢？先拿首页中显示的“SIEM：TOP 10 Events by Product”这种饼图进行解释，从表面上看，这种饼图在数据可视化方面表现出色，但在一幅饼图中若有多个相似值，大多数人无法准确区分它们的差异，除非将鼠标移动到扇面之上时，会显示报警类型及数量大小，而且饼图没有明显的开始和结束标志，这使得人们难以集中注意力关注某一类报警，尤其是饼图在表达多种类型报警时，各个扇面之间很难保持高对比度（实际上是图论中学过的扇面的着色问题），但柱状图却能避免这些问题，这些柱状图，往往以坐标的方式提供参考系，有 X 轴和 Y 轴，可以有效而准确地将数据展现给人们。如图 9-18 所示。

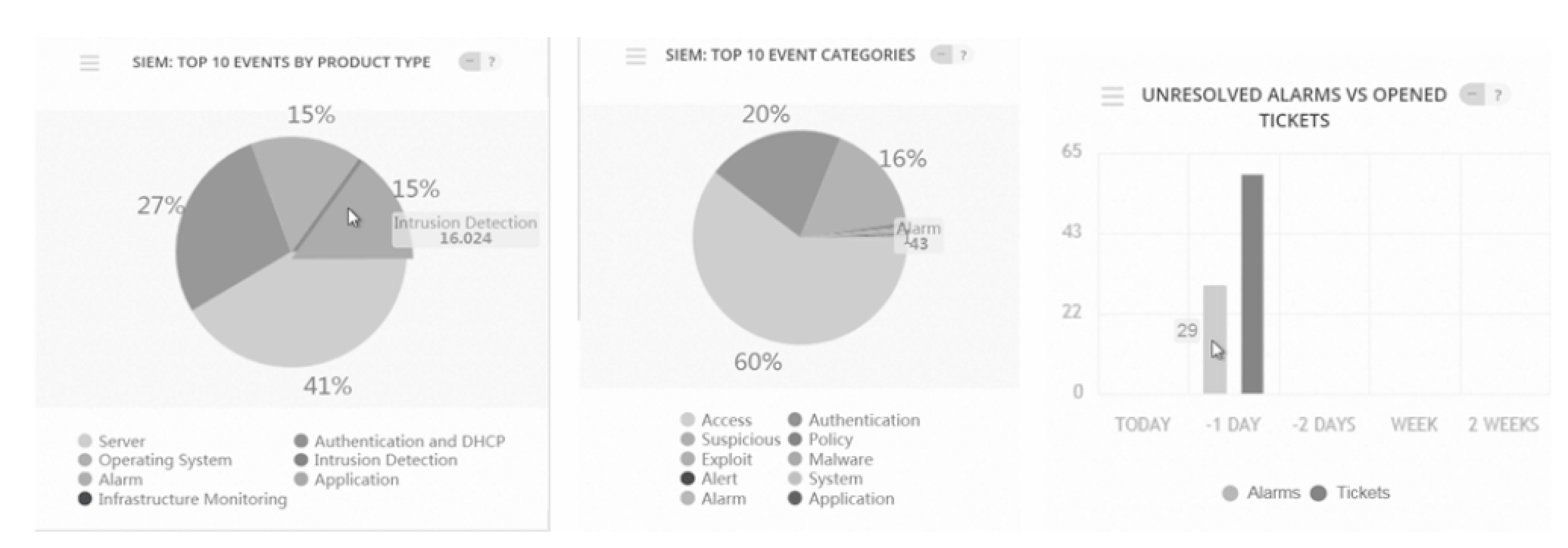


图 9-18　饼图与柱状图表达方式对比

（3）迷你图

迷你图是一种文字大小般的图形，它们可以绘制出迷你的折线图和迷你的柱状图，如图 9-19 所示。在 Environment Snapshot 系统环境快照、流量抓包以及漏洞扫描等界面中都能发现它的身影，在这种迷你折线图中，图形中最高点和最低点都添加了数值，能够节省空间，也能表达准确的、准实时的内容。

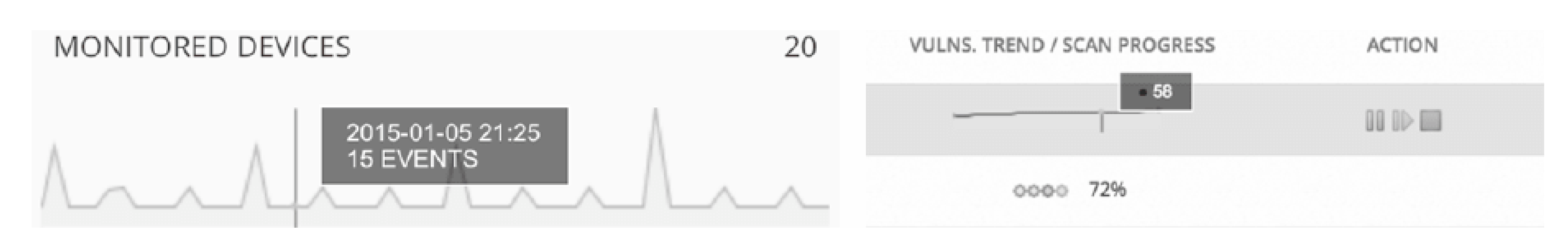


图 9-19　OSSIM 中使用的迷你图

9.2 OSSIM 的 Web UI 菜单结构

了解 OSSIM Web 的站点目录有利于开发 Web 界面，下面基于 OSSIM 4.8 版本予以介绍。对于 OSSIM 4.2 版的站点目录结构在《UNIX/Linux 网络日志分析与流量监控》一书中 14.5 节，OSSIM 4.8 的 Web 目录结构如表 9-1 所示。

表 9-1　OSSIM 4.8 站点目录结构

一级菜单	二级菜单	调用页面
DASHBOARDS	Overview	./dashboard/index.php
	Deployment status	./deployment/index.php
	Risk Maps	./risk_maps/view.php
	OTX	./reputation/index.php
Analysis	Alarms Group View	./alarm/alarm_console.php ./alarm/alarm_group_console.php
	Security Events（SIEM） Real Time	./forensics/base_qry_main.php ./control_panel/event_panel.php
	Raw Logs	./sem/index.php（免费版不提供）
	Tickets	./incidents/index.php
ENVIRONMENT	Assets Asset Discovery	./assets/index.php ./netscan/index.php
	Groups&Networks Network Groups	./assets/list_view.php ./netgroup/netgroup.php
	Vulnerabilities	Overview: ./vulnmeter/index.php Scan Jobs:./vulnmeter/manage_jobs.php Settings:./vulnmeter/webconfig.php Threat Database:./vulnmeter/threats-db.php
	Profiles	./ntop/index.php
	NetFlow	./nfsen/nfsen.php
	Traffic capture	./pcap/index.php
	Availability	./nagios/index.php
	Detection	./ossec/status.php Agents:./ossec/agent.php Agentless:./ossec/agentless.php Edit Rules:./ossec/index.php Config：./ossec/config.php Ossec control:./ossec/ossec_control.php Wireles IDS:./wireless/index.php

（续表）

一级菜单	二级菜单	调用页面	
REPORTS		Alarms Report 生成文件:./report/os_reports/Alarms/general.php Business&Compliance ISO PCI Report 生成文件: ./report/os_reports/BussinessAndComplianceISOPCI/general.php Tickets Status Report 生成文件:./report/os_reports/Tickets/general.php SIEM Events 生成文件：./reports/os_reports/Siem/general.php Vulnerabilities Report 生成文件：./vulnmeter/lr_respdf.php	
CONFIGURATION	Administration	USERS	./session/users.php Activity:./conf/userlog.php
		MAIN	./conf/index.php
		BACKUP	./backup/index.php
	Deployment	Alienvault Center:./av_center/index.php Sensors:./server/sensor.php Servers:./server/server.php	
		Scheduler:./av_inventory/index.php	
		Locations:./sensor/locations.php	
	Threat Intelligence	Policy:./policy/policy.php Edit pPolicy Groups:./policy/policygroup.php	
		Actions:./action/action.php	
		Ports:./port/port.php Port Groups:./port/portgroup.php	
		Directives:./directives/index.php	
		ComplianceMapping:./compliance/iso27001.php PCIDSS2.0:./compliance/pci-dss.php Run Scripts:./compliance/mod_scripts.php	
		Cross Correlation:./conf/pluginref.php	
		Data Source:./conf/plugin.php Data Source Groups:./policy/plugingroups.php	
		Taxonomy:./conf/category.php	
		Knowledge Base:./repository/index.php	
SETTINGS	My Profile	./session/user_form.php	
	Current Sessions	./userlog/opened_sessions.php	
	User Activity	./userlog/user_action_log.php	
Support	Help	./help/index.php	
	Downloads	./downloads/index.php	

网站根目录路径位于/usr/share/ossim/www/。

9.3 OSSEC 架构与配置

9.3.1 OSSEC 架构

OSSEC 是一款开源的入侵检测系统，采用了客户端/服务器模式，客户机通过客户端程序将数据发送到服务器端分析，主要功能包括日志分析、完整性检查、rootkit 检测等。OSSEC 被集成到 OSSIM 平台之后，除了具有日志分析、入侵检测系统功能外，还添加了日志关联分析报警功能。当客户机通过客户端程序，将数据发回服务器端进行关联分析之后，把结果存入数据库并发出报警。OSSIM 中集成的 OSSEC 和单独安装的 OSSEC 系统最大的区别就是增加了关联分析引擎，其架构如图 9-20 所示。

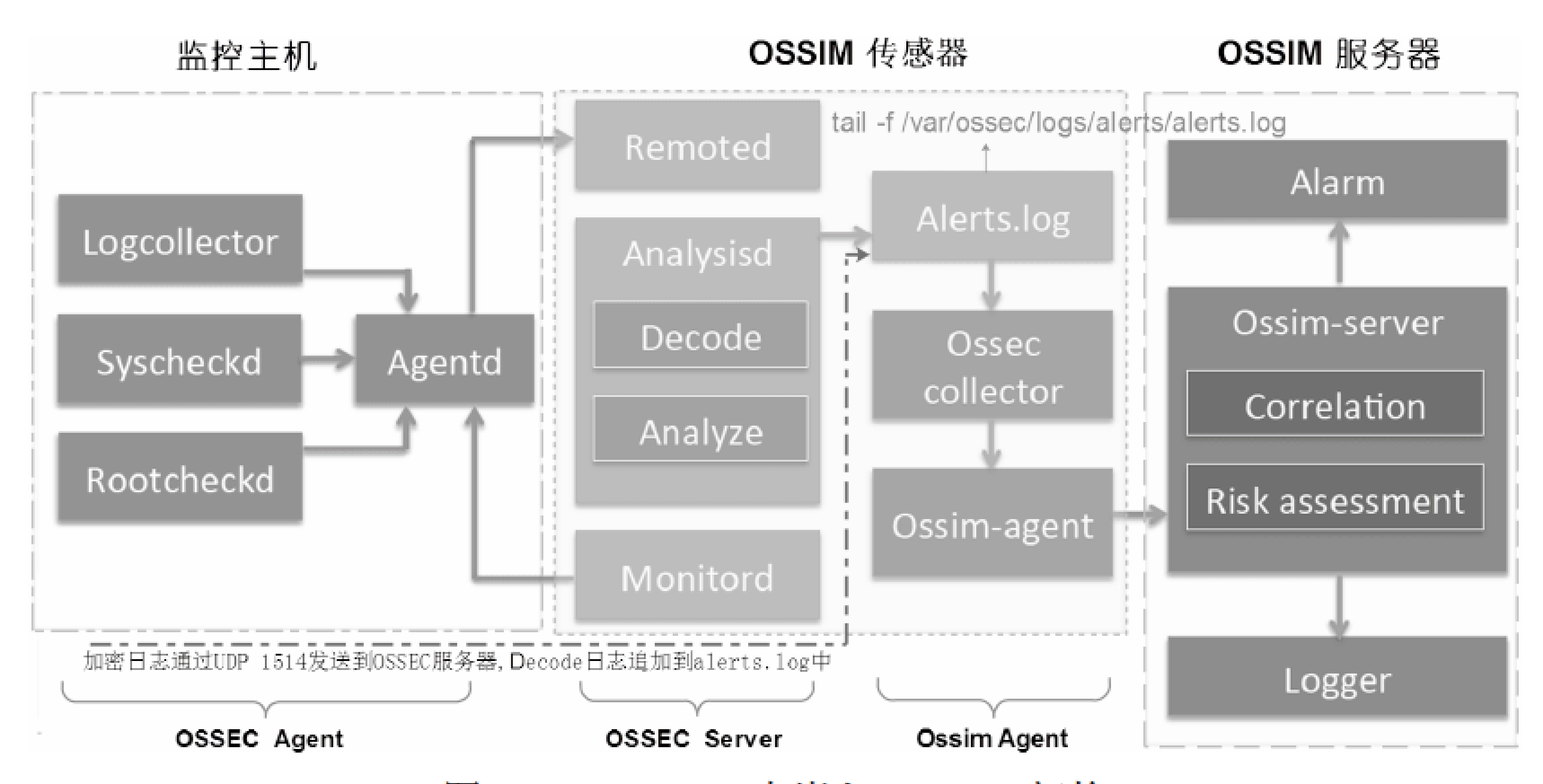


图 9-20　OSSIM 中嵌入 OSSEC 架构

从图 9-20 可知 OSSEC 的事件报警流程：Agentd → UDP 1514 → Remoted → Analysisd → Aerts.log → Ossim-agent→ Correlation → Alarm。OSSIM 中的 OSSEC 模块整体分为 Agent、Server 和关联分析模块，下面将详细分析各进程的功能。

9.3.2 OSSEC Agent 端进程

下面以 Linux 平台为例，Agent 主要包含以下几个进程：

（1）logcollector，读取从 syslog、wmi 收集的日志，进程名称为 ossec-logcollector。

（2）syscheckd，文件系统完整性检查（File Integrity Check）、Windows 注册表完整性检查，进程名为 ossec-syscheckd。它是 OSSEC 内部完整性检测进程的名称，它周期性检查是否有任何配置文件（或 Windows 注册表）发生改变。当系统的完整性被改变时，它能够通过比较文件的 MD5 校验和发现问题。

Syscheck 工作原理：代理默认每 6 小时（在/var/ossec/etc/ossec.conf 文件的<frequency>字段定义，21600 秒表示频率为 6 小时）扫描一次系统，并发送校验和（checksum）到 Server 端。Server 端存储这些校验和文件，并比对他们的差异。如果发现任何改变，将会发送报警。

数据存放位置在/var/ossec/queue/syscheck/。

在 ossec.conf 配置文件里的 syscheck 项，提供了一个将要被监控的目录（/etc、/usr/bin、/usr/sbin）和文件（/root/myfile、/var/src/sys），check_all 选项会检查文件的 md5、sha1、owner 和权限，对 ossec.conf 相应字段进行如下修改。

```
<syscheck>
    <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/root/myfile,/var/src/sys</directories>
</syscheck>
```

如果不想监控某些文件和目录，可以用 ignore 选项。下面看一个系统自带的例子，还是打开/var/ossec/etc/ossec.conf 配置文件，修改如下内容：

```
<syscheck>
    <ignore>/etc/mydir</ignore>
    <ignore>/root/mydir</ignore>
    <ignore type="sregex">.log$|.tmp</ignore>
</syscheck>
```

在 ignore 选项中，该类型属性可以设置为 sregex，指定一个正则表达式语法。

```
<syscheck>
    <ignore type="sregex">^/var/log/applog</ignore>
</syscheck>
```

本地规则可以修改需要更改的指定文件和目录。

```
<rule id="100345" level="12">
    <if_matched_group>syscheck</if_matched_group>
    <match>/var/www/docs</match>
    <description>Changes to /var/www/htdocs - Critical file!</description>
</rule>
```

在上面的例子中，当/var/www/docs 目录中的文件发生改变时，将会产生一个严重的报警。在 Linux 和 Windows 中，OSSEC 支持实时监控文件完整性检查，directories 选项可以指定要监控的文件和目录，并设置 realtime="yes "。

```
<syscheck>
<directories realtime="yes"
check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin</directories>
</syscheck>
```

在该示例中，目录/etc、/usr/bin 、/usr/sbin 会被实时监控。

实时监控并不会立即开始，首先 ossec-syscheckd 需要扫描文件系统，并添加每一个子目录到实时队列。完成这些工作约 30 分钟。实时监控仅对目录有效，可以监控 /etc 或者 Windows 系统下的 C:\Program Files\目录，但不能为单独的文件。

OSSEC 系统可比较文件的变化，OSSEC 支持发送比较报告功能，配置 syscheck 显示文件比较的不同很简单，只要添加 report_changes="yse"到 directories 选项即可。

```
<syscheck>
   <directories report_changes="yes" check_all="yes">/etc</directories>
   <directories check_all="yes">/bin,/sbin</directories>
</syscheck>
```

让 syscheck 停止对系统进行扫描，只需在 ossec.conf 中指定<scan_on_start> 选项为"no " 即可。

（3）rootcheckd：用来监控流氓软件、监控所有网络接口和 rootkits 检测，进程名为 rootcheckd。

OSSEC HIDS 会在每个安装代理的系统中运行 rootkit 检测。Rootkit 每 120 分钟会探测任何可能已经安装的 rootkit 和日志分析及安全性检测，OSSIM 系统中集成的 OSSEC HIDS 将在目录/var/ossec/etc/shared/中读 rootkit_files.txt 文件，其中包含了使用文件和 rootkit 的基本数据。

我们使用所有这些系统调用，有一些内核级 rootkits 从系统调用中隐藏了一些文件。

```
#/var/ossec/bin/ossec-control status
ossec-monitord is running…
ossec-logcollector is running…
ossec-remoted is running…
ossec-analysisd is running…
ossec-maild not running…
ossec-execd not running…
```

ossec-control 显示状态也可以在 Web UI 下查看，路径如图 9-21 所示。

DETECTION
HIDS　WIRELESS IDS
OVERVIEW | AGENTS | AGENTLESS | EDIT RULES | CONFIG | HIDS CONTROL
HIDS CONTROL　HIDS LOG　ALERTS LOG
Select sensor: VirtualUSMAllInOne [192.168.11.150]
▸ Hide actions
ACTIONS
Client-syslog is NOT running　ENABLE
Agentless is NOT running　ENABLE
Debug is enabled　DISABLE
HIDS service is UP　STOP　RESTART
HIDS OUTPUT

```
monitord is running...
logcollector is running...
remoted is running...
syscheckd is running...
analysisd is running...
maild not running...
execd not running...
```

图 9-21　查看 OSSEC 运行状态

由于默认情况下 OSSEC 没有配置 Client-syslog、Agentless 以及邮件，所以这些服务并没有启动，这属于正常显示。

（4）agentd：把数据加密转发到服务器，进程名为 ossec-agentd。

9.3.3　OSSEC Server 端

OSSEC Server 端进程包括以下内容：

（1）remoted：接收由代理进程发送的数据，采用 UDP 协议，通信端口为 1514，进程名称为 ossec-remoted。在排除日志故障时，可启动 ossec-remoted 进程的 DEBUG 模式，操作如下：

```
#/var/ossec/bin/ossec-remoted -d
```

同时查看日志：

```
#tail -f  /var/ossec/logs/ossec.log
```

（2）analysisd：数据处理主模块，进程名称为 ossec-analysisd，包含以下功能：

- 日志预解码（提取系统日志头中主机名和程序名以及时间等）；
- 日志解码（利用正则表达式提取敏感关键字，比如提取 IP、用户名、ID 等）；
- 日志分析（检测日志解码后的相关匹配规则）。

（3）monitord：存活监控，进程名称为 ossec-monitord。

在 OSSIM 的 Web UI 中，我们如何了解 HIDS 工作情况呢？可通过单击 ENVIRONMENT → DETECTION 菜单实现，打开界面如图 9-22 所示。

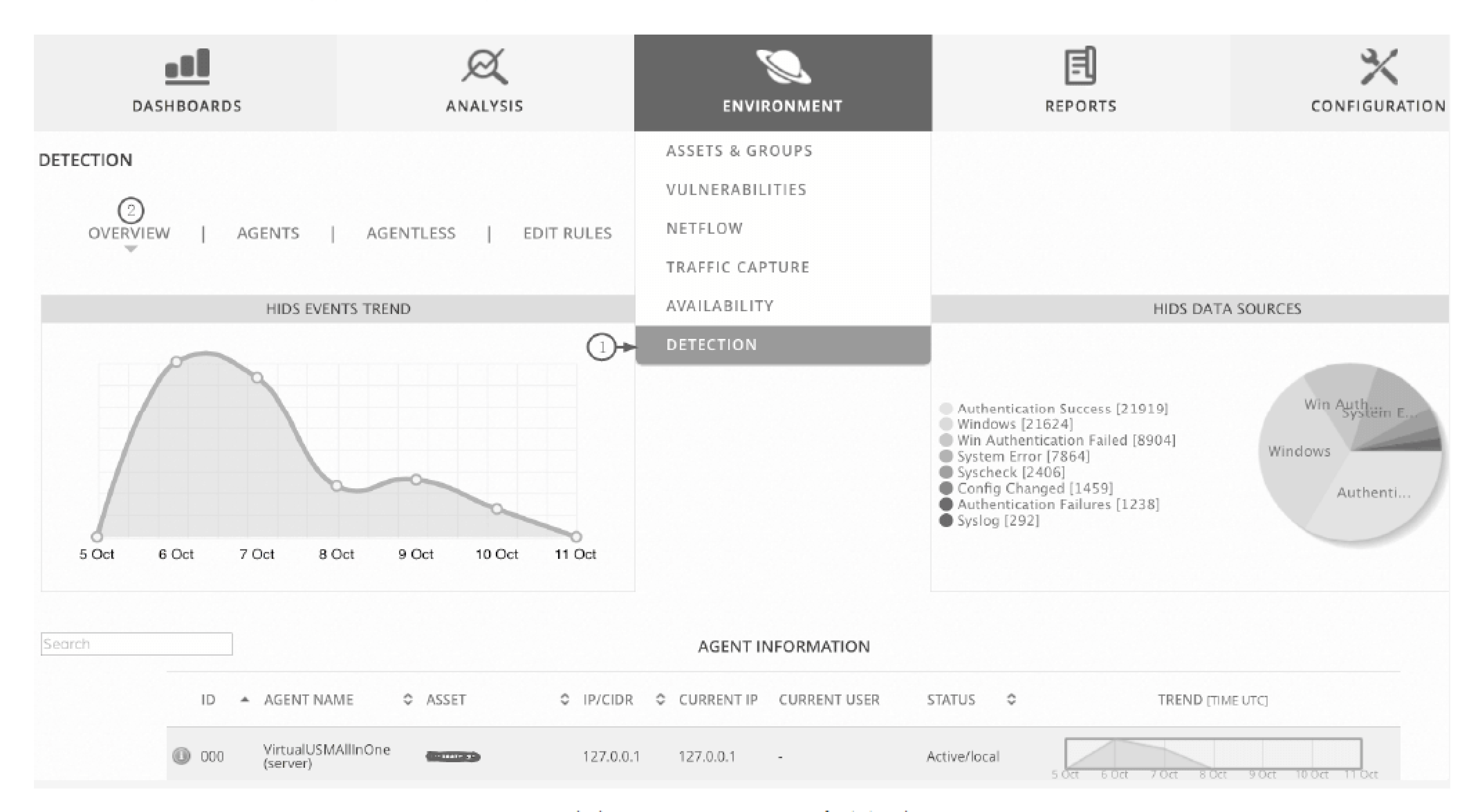


图 9-22　OSSEC 主界面

从上图可以发现 AGENT、EDIT RULES、CONFIG、HIDS CONTROL 模块中还有子模块，各模块关系总结如图 9-23 所示。

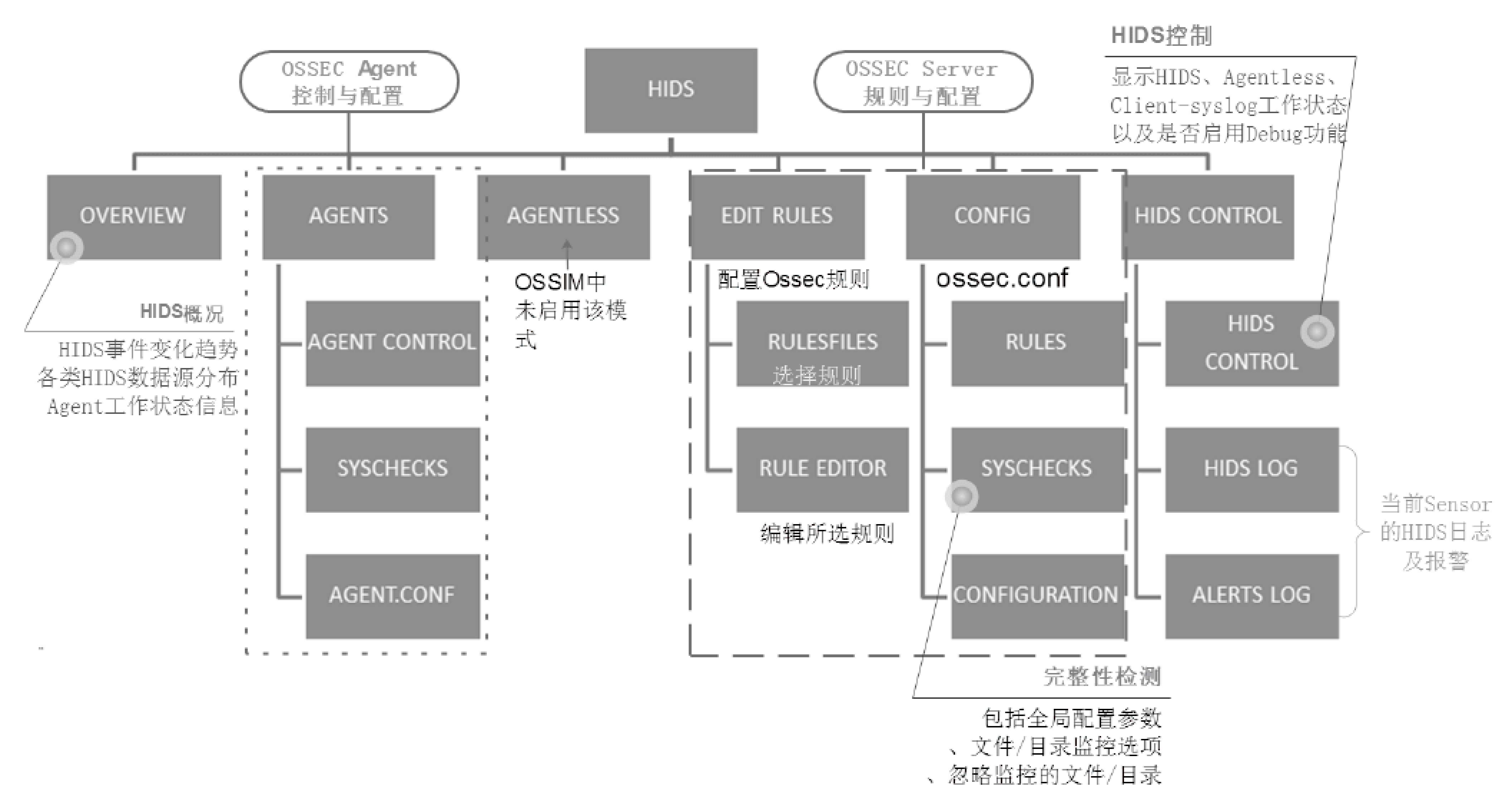


图 9-23　OSSEC 子模块菜单结构

图 9-23 不但为我们提供了理解 HIDS 系统的一个清晰思路，而且反映了信息：

- 从整体上看，与 Ossec Agent 相关的配置在图的左侧，与 OSSEC Server 相关的配置在右侧。在维护 HIDS 时，通过点击对应模块就能准确找到对应模块，而不会被众多菜单弄得没有头绪。比如需要对 Agent 下的配置文件进行管理，可选择 AGENTSN→AGENT.CONF，在图 9-23 中，点线框中的项目 AGENT.CONF 中操作。
- 如果需要编辑 OSSEC Server 的规则文件，则在 EDIT RUlES 列表中选择一个规则，然后再进行编辑。OSSEC 所有规则存放在/var/ossec/rules/，为了防止误操作，在该目录中还有由 backup-rules 开头的子目录，顾名思义起到还原最初设置的目的。
- 通过 HIDS 控制台能及时发现停止的服务以及日志。

9.3.4　OSSEC 配置文件和规则库

多数服务的配置文件在/etc/，但 OSSEC 服务配置文件位于/var/ossec/etc/ossec.conf 中。通过在 Web UI 单击 Environment→Detection→HIDS 中的 CONFIG 选项，通过图形界面就能轻松实现 OSSEC 规则的增减，如图 9-24 所示。

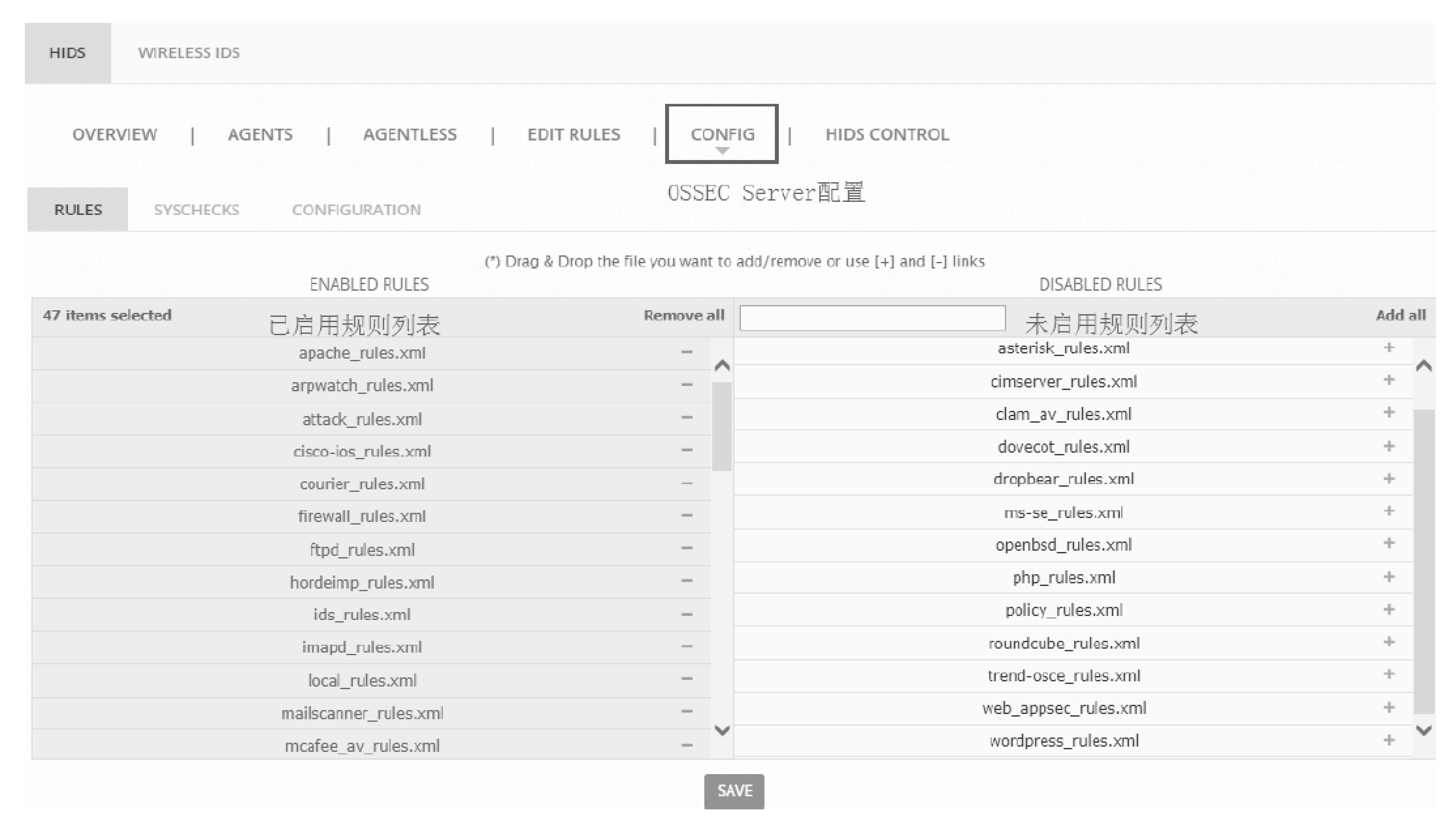


图 9-24　管理 OSSEC 规则

OSSEC 能通过分析日志产生报警，主要因为内置大量的 OSSEC 规则（约 990 条），规则库位于/var/ossec/rules/*.xml。若要加载规则文件，需要在/var/ossec/etc/ossec.conf 中配置，默认的配置如下：

```
<ossec_config>
  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
    <include>telnetd_rules.xml</include>
    <include>syslog_rules.xml</include>
    <include>arpwatch_rules.xml</include>
     ......
    <include>clam_av_rules.xml</include>
    <include>bro-ids_rules.xml</include>
    <include>dropbear_rules.xml</include>
    <include>local_rules.xml</include>
</rules>
</ossec_config>
```

上面配置是逐条加载规则，读者也许会认为一次加载所有的规则文件不是更方便吗？其实不然，很多情况下我们需要自定义规则，而且有些过时的规则也需要及时移除。如果一次性将所有规则全部加载，则无法满足我们的需求，故笔者推荐大家使用逐条添加的策略。

9.3.5　测试规则

当自己定制规则或修改已有的OSSEC规则时，都需要进行反复测试，第2章讲过regexp.py工具的使用，我们先利用regexp.py对ossec-single-line.cfg插件进行验证，如图9-25所示。

```
localhost:~# /usr/share/ossim/scripts/regexp.py /var/ossec/logs/alerts/alerts.log /etc/ossim/agent/
plugins/ossec-single-line.cfg v
Multiple regexp mode used, parsing /etc/ossim/agent/plugins/ossec-single-line.cfg
Matched using 0025 - syslog errors
AV - Alert - "1444536011" --> RID: "1002"; RL: "2"; RG: "syslog,errors,"; RC: "Unknown problem some
where in the system."; USER: "None"; SRCIP: "None"; HOSTNAME: "localhost"; LOCATION: "/var/log/sysl
og"; EVENT: "[INIT]Oct 11 00:00:10 localhost nfcapd[2620]: Ident: 'EBFCC3E41C8942C8ABA8B6DB0A0952F1
' Flows: 3, Packets: 13, Bytes: 6208, Sequence Errors: 0, Bad Packets: 2[END]";
```

图 9-25　测试 ossec 插件

除此之外OSSEC系统还提供了ossec-logtest工具，它通过日志告诉用户将如何解码，以及哪些信息被提取和触发了哪一条规则。比如，我们需要监控Apache服务的访问日志，将它添加到OSSEC日志源，修改Agent端的ossec.conf配置文件，加入如下配置：

```
<localfile>
<location>/var/log/apache2/access.log</location>
<log_format>apache</log_format>
</localfile>
```

标签<location>的内容为需要监控服务的访问日志。接下来，在命令行接着使用/var/ossec/bin/ossec-logtest 命令，通过返回值判断修改或新添加的规则是否匹配成功，待成功之后再重启OSSEC Server和Agent端的服务，这样更加稳妥。

对日志进行分析，通过如图9-26所示的命令完成。

```
VirtualUSMAllInOne:~# cat /var/log/auth.log |/var/ossec/bin/ossec-logtest -a
2015/10/04 00:08:33 ossec-testrule: INFO: Reading local decoder file.
2015/10/04 00:08:33 ossec-testrule: INFO: Started (pid: 2532).
** Alert 1443931713.1: mail  - syslog,attacks,invalid_login,
2015 Oct 04 00:08:33 VirtualUSMAllInOne->stdin
Rule: 40101 (level 12) -> 'System user successfully logged to the system.'
User: daemon
Oct  4 00:01:30 VirtualUSMAllInOne su[1999]: + ??? root:daemon
```

图 9-26　ossec-logtest 测试结果

理解这行命令后，我们稍加扩展成为如下命令：

```
#cat /var/ossec/logs/alerts/alerts.log  | /var/ossic/bin/ossec-logtest -a
```

其他测试命令还包括：

```
#/var/ossec/bin/ossec-logtest  -t   /*测试配置*/
#/var/ossec/bin/ossec-logtest  -d   /*运行调试模式*/
#/var/ossec/bin/ossec-logtest  -h  /*显示帮助信息*/
```

9.3.6　分布式系统中 OSSEC Agent 的管理

OSSIM能够非常方便地实现分布式部署，相关内容在第2章就已经详细讲解，这里为大

家介绍分布式环境下，OSSEC Agent 管理需要注意的内容。

OSSIM 中以各个网段的 Sensor 为单位来管理监控网段内的所有 Agent，所以在分布式 OSSIM 环境中，安装、使用 Agent 时，需要格外注意 Sensor 所监管的各个 Agent。一个简单的分布式环境，图 9-27 所示。

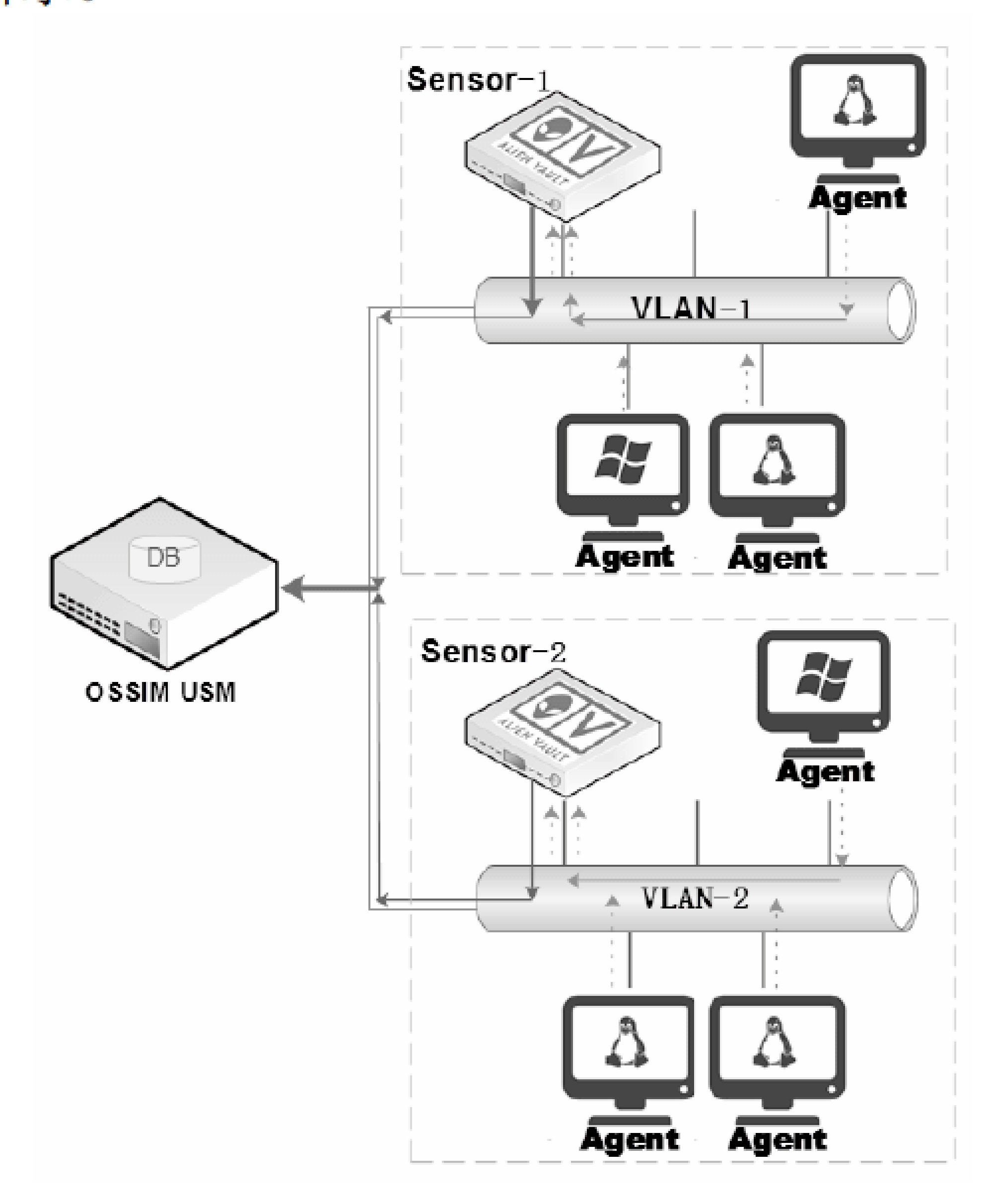


图 9-27　分布式 OSSEC 系统 Agent 配置数据流向

在上图中 Sensor-1，这台主机内本身装有 Ossec Agent，同时还统一管理监控网段内的所有 Windows、Linux 的 Ossec Agent 的配置文件。举个例子，大家在为 VLAN-1 中的一台主机安装 Agent，首先要选择负责监控 VLAN-1 的 Sensor-1，如图 9-28 所示。这里假设 Sensor（192.168.11.188）选择之后，系统会列出当前所有 Agent，这时再单击添加 Agent 按钮，才是正确的安装方法。

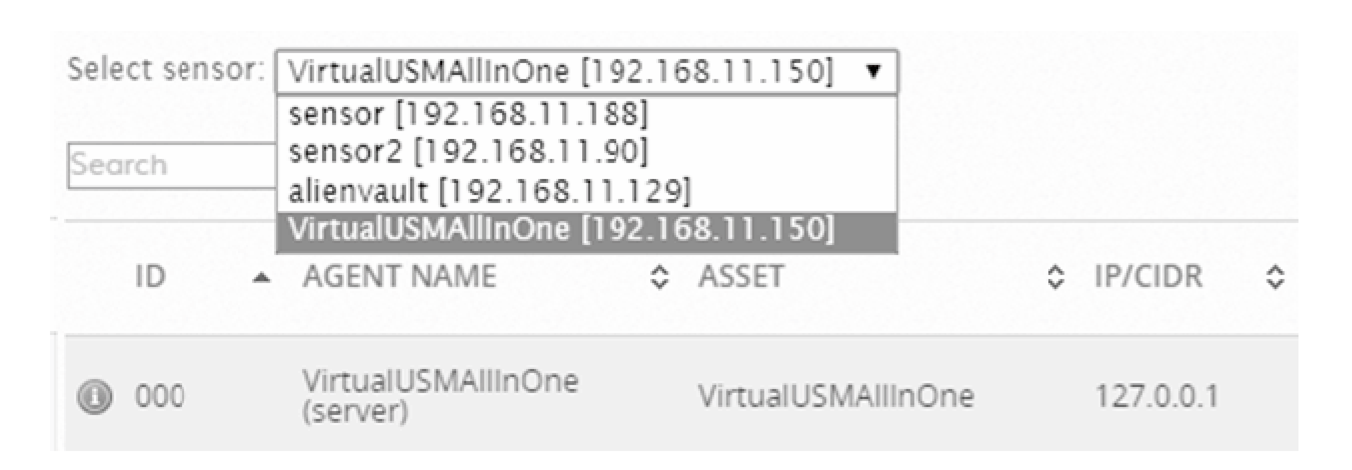


图 9-28　多 Sensor 环境添加 Agent

9.3.7　OSSEC 日志存储

首先确保磁盘分区有足够剩余空间，OSSEC 把日志存储在/var/ossec/logs/alerts 目录，在基于日志事件生成的年份和月份的目录中保存日志。例如/var/ossec/logs/alerts/年/月/ ossec_alerts_日期.log，如图 9-29 所示。

图 9-29　日志输出

与 OSSIM 中的 RAW LOG 日志一样，OSSEC 日志也是文本格式，这意味着可以直接查看，支持 Perl 脚本检索。在关联分析之后，OSSIM 把重要的 OSSEC 报警按照一定格式转储到 MySQL 数据库（alienvault）中以便今后分析，这才是最重要的数据。另外，我们可以在 SIEM 控制台中，通过数据源筛选快速查看 HIDS 事件，细节下面再介绍。

9.3.8　OSSEC Agent 安装

我们知道 OSSEC 属于 C/S 架构，由于 OSSEC Server 安装在 OSSIM 系统中并已配置，我们通过 OSSEC Server+Agent 方式实现 HIDS 功能。我们先在 Windows 系统上安装 Agent，然后在 Linux 平台安装。

目前在 OSSIM 4.8 的 Web UI 中安装 OSSEC Agent 更为方便，因为系统会把和用户交互过程中的配置信息自动写入配置文件。下面分为单独 Agent 安装和自动化安装 Agent 两种方式来介绍。

1. Windows 系统上安装 Agent

（1）单独 Agent 安装

接下来以 Windows 7 系统为例来添加 OSSEC Agent。为了调试方便，调试程序暂时关闭 Windows 7 系统防火墙，该实验选用 OSSIM 4.8 版本，此处 Agent 所在系统的 IP 为 192.168.11.2，OSSIM Server IP=10.32.11.150。

进入 Web UI 界面，依次选择 ENVIRONMENT→DETECTION→HIDS→AGENTS，如图 9-30 所示。然后选位于屏幕左下方的“ADD AGENT”按钮。

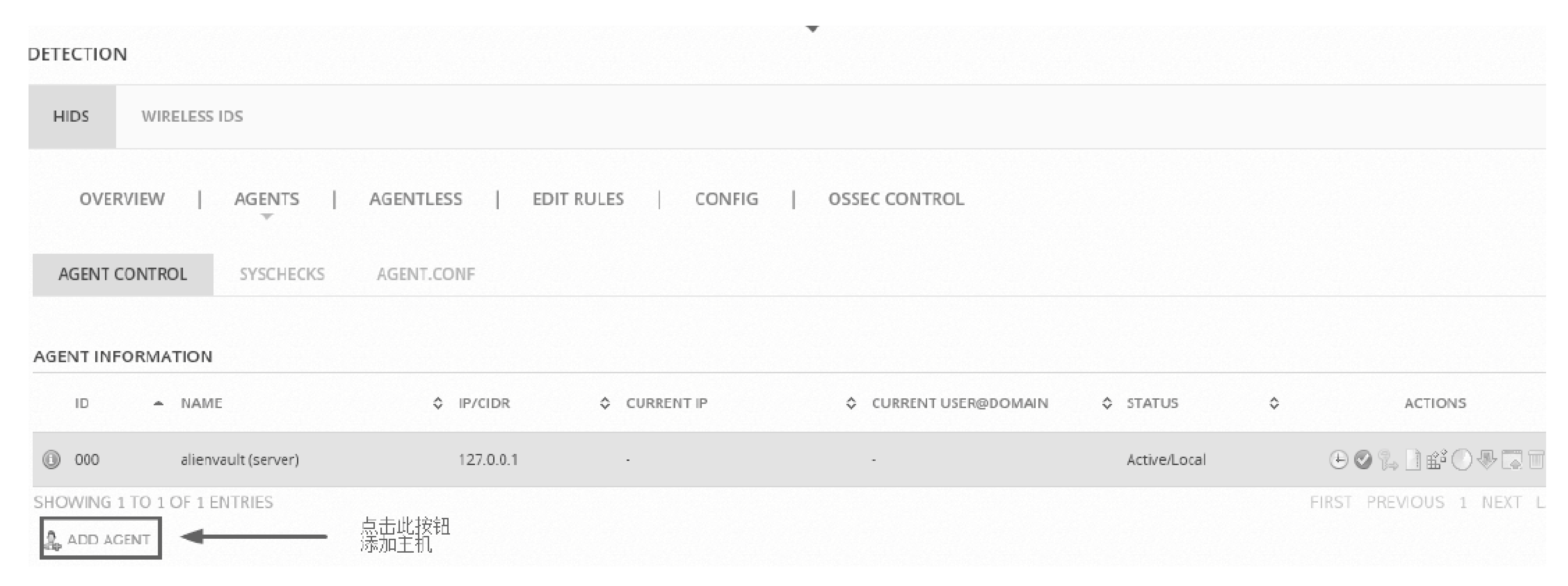
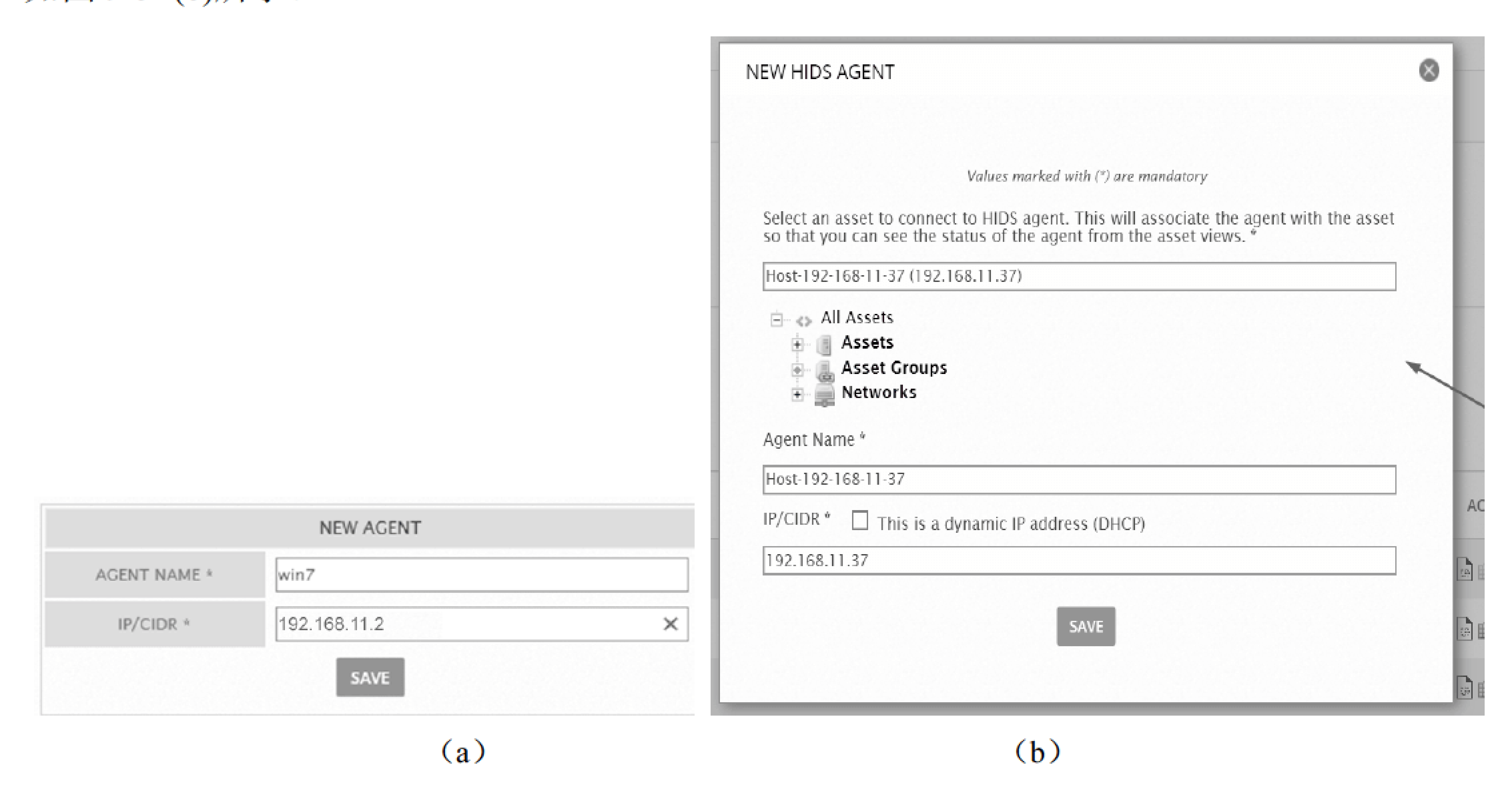


图 9-30　添加代理

接着输入客户机名（比如可以是这台机器的 NETBIOS 名“Win7”IP 地址 192.168.11.2，在 OSSIM 4.8 系统中添加 Agent 界面，如图 9-31（a）所示，而在 OSSIM 5.0 之后的版本中添加 Agent，不用手工输入 IP 和机器名称，而直接在资源池中选取，更加方便用户，操作界面如图 9-31(b)所示。



（a）　　　　（b）

图 9-31　设置代理名称和 IP

在进行 Agent 各种操作时有 9 个按钮每一个功能都需要大家掌握，其含义如图 9-32 所示，为了下载预配置 Agent 代理程序，我们先单击第 7 个按钮，即可开始下载（大小约为 792KB），完成后双击安装即可。注意：此处下载的程序中包含了配置文件和 Key，专门为所选定的客户机定制，复制到其他机器无法使用，这个和在 http://www.ossec.net/网站上下载的通用代理程序不同。

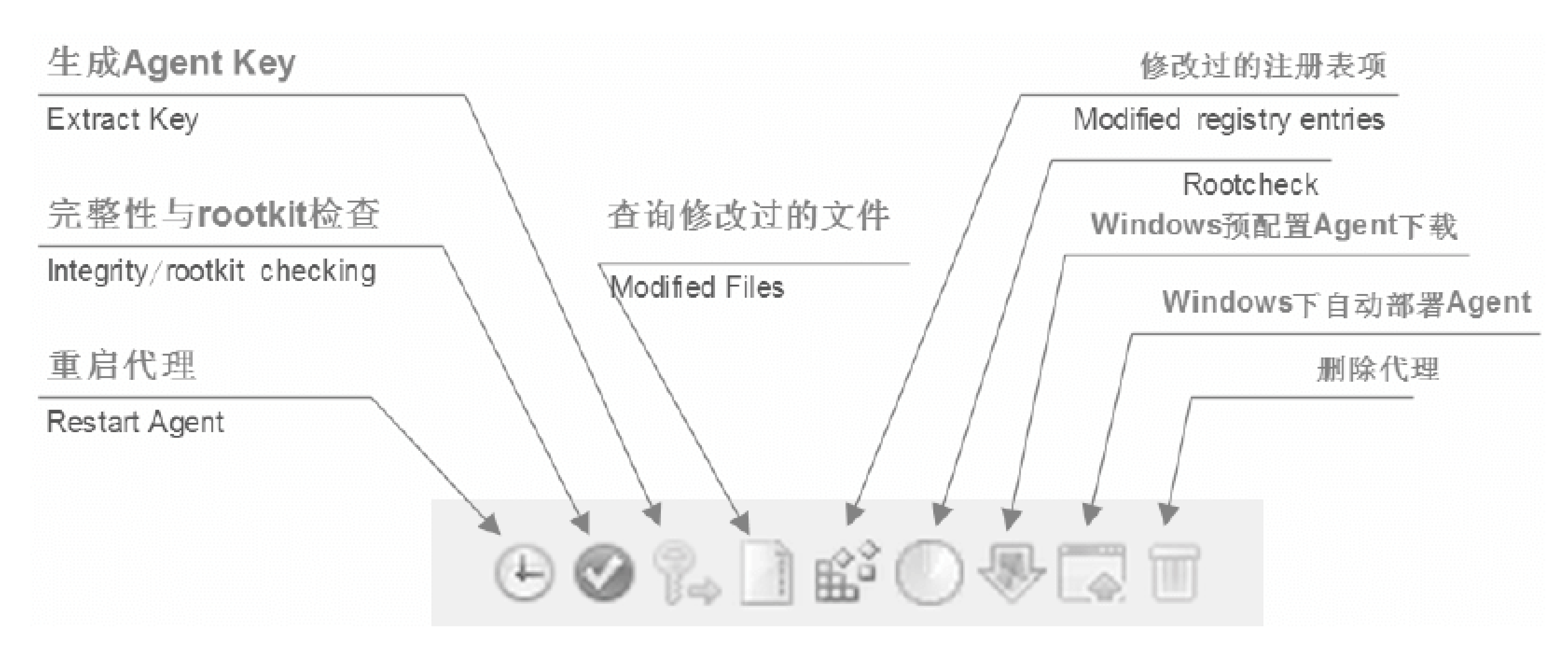


图 9-32　Agent 操作按钮含义对照

Agent 软件的安装路径为 C:\Program Files(x86)\ossec-agent\。依次选择 Windows 系统的开始菜单→ossec→Manage Agent 程序（程序名称为 win32ui.exe），因为是安装的预配置程序，所以这时会发现 OSSEC Server IP 地址和认证密钥都可自动填写完毕，如图 9-33 所示。还可以在 Web UI 中，单击如图 9-32 所示的第 3 个按钮，查看生成的密钥。

图 9-33　在代理程序中查看 OSSEC 服务器 IP 地址及客户端密钥

如使用旧版本（OSSIM 4.3+OSSEC 2.7），则需要手动将 Web UI 生成的密钥复制至客户端的验证密钥框中并保存，最后启动代理程序。

Sensor 仅用来管理它所监控网段内的主机，不能跨越路由，而 Sensor 和 OSSIM Server 之间的通信却可以跨越多个路由，Sensor、OSSIM Server 可以在同一个 IDC 机房，也可分别在异地机房。在实践中有些读者会出现下面示例中的错误。

在母机安装有 Windows 系统（IP 地址为：10.32.x.y），安装 Vmware，在其中安装了两台虚拟机分别是 192.168.91.228（安装 OSSIM USM）和 192.168.91.140（安装 Windows 7 系统）。能够看出 Sensor 和 192.168.91.140 都在一个网段，而母机和 Sensor 之间，不在同一网段。此时在 OSSIM Web UI 中部署 HIDS，就会出现同一个网段内的客户机都能安装 HIDS，而不在一个网段的无法自动安装，如果手动安装却无法正常通信，如图 9-34 所示。

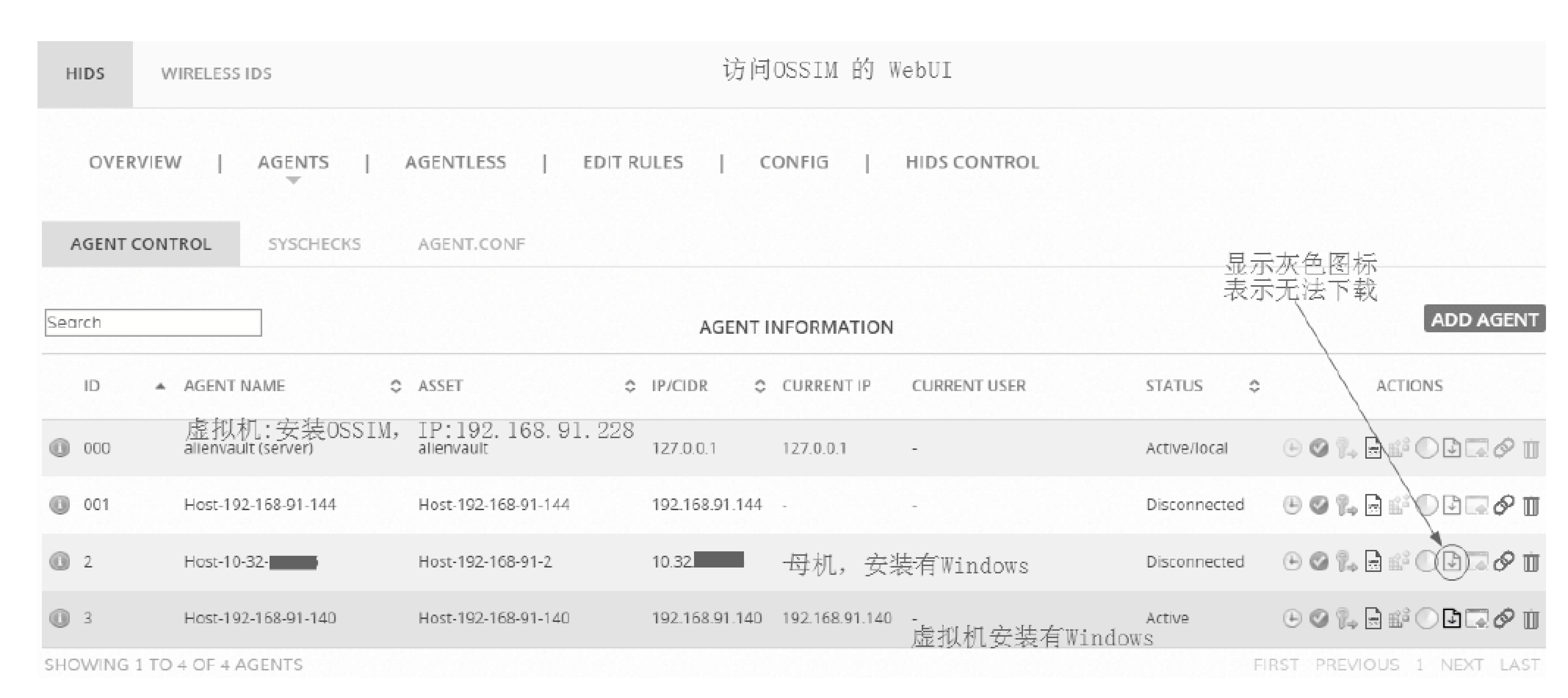


图 9-34　无法下载 Agent 实例

（2）检查安装情况

客户机浏览器中打开 OSSIM 管理界面，在 Analysis→Detection→HIDS 中，可以查看到代理已成功添加，状态为 Active，如图 9-35 所示。将鼠标移动到 ID 号前面“！”号位置，便能查看当前 Agent 的工作状态。

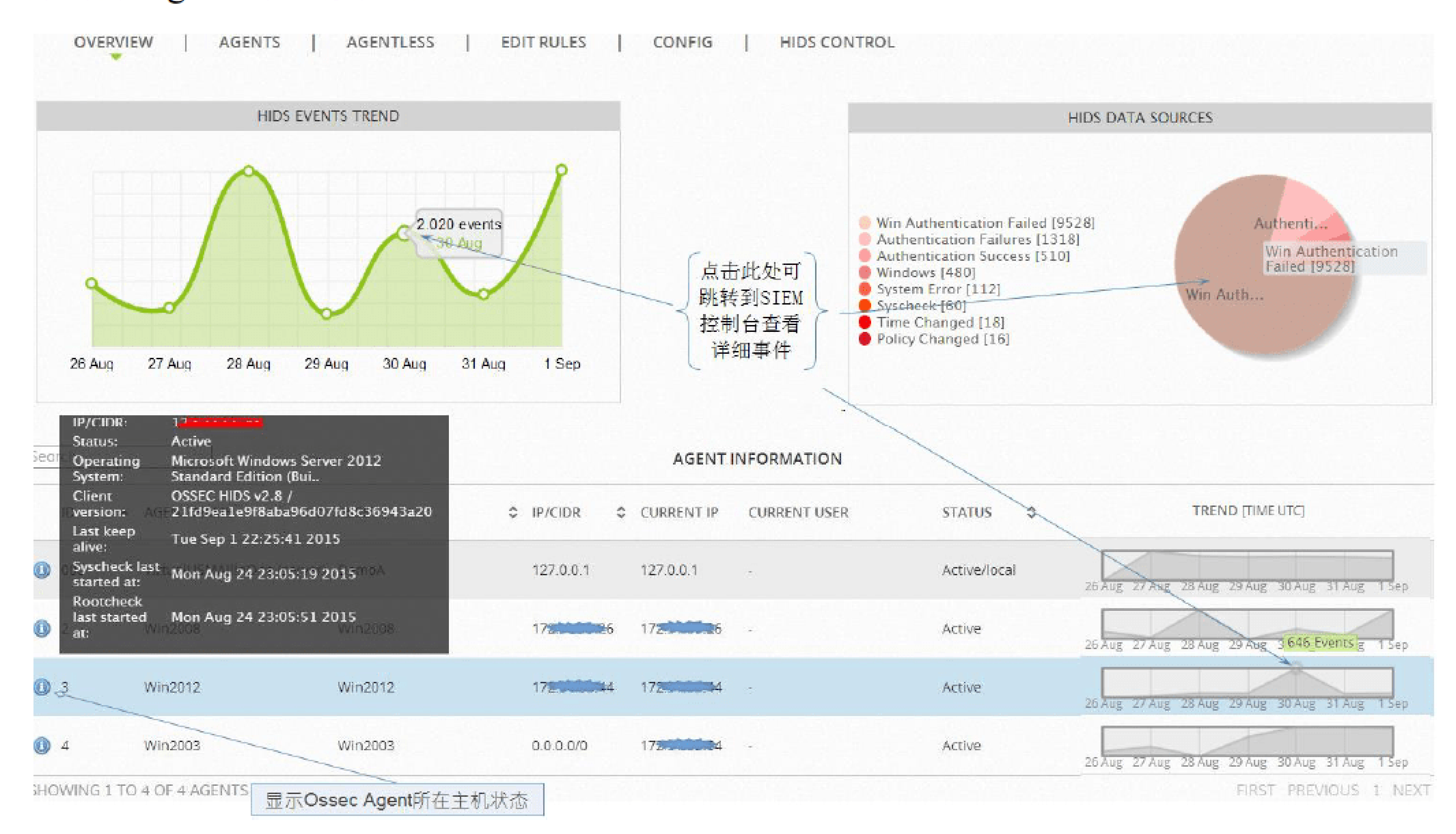


图 9-35　HIDS 中 Agent 的工作状况

图中显示了所有活动代理的情况. 在右侧“HIDS DATA SOURCES”区域的扇形显示图像中，当鼠标移动到扇形区域上，会立刻显示出当前 OSSEC 日志名称及所占比例，单击该区域会调出 SIEM 中显示 OSSEC 的事件。另外在命令行下操作时还可以知道更详细的信息，我们在命令行中输入以下命令：

```
# /var/ossec/bin/agent_control -lc
```

由上图的显示结果可以判断，OSSIM 代理（Agent）程序扮演着检测引擎的角色，它根据主机行为特征库对受检测主机上的可疑行为进行采集、分析和判断，并把警报信息发送给控制端程序，特征库包括很多类操作系统上的事件。这些事件检查可疑的文件传输，以及被拒绝的登录企图。特征库也可包括来自许多应用程序和服务，如 Secure Shell、Sendmail、Bind 和 Apache 服务等。HIDS 可以根据结果来进行判断，比如关键系统文件有无在未经允许的情况下被修改，包括访问时间、文件大小和 MD5 密码校验值。

（3）接收 Windows 事件

当代理装好之后，接下来就要查看从 Agent 端发送到 Ossec Server 端的 Windows 事件，OSSIM 的 Web UI 中提供非常友好的界面，获取这些信息还是在 SIEM 控制台中进行。在图 9-36 展示出代理所在客户机 192.168.11.2 这台 Windows 机器上发出的一条审计失败的事件。

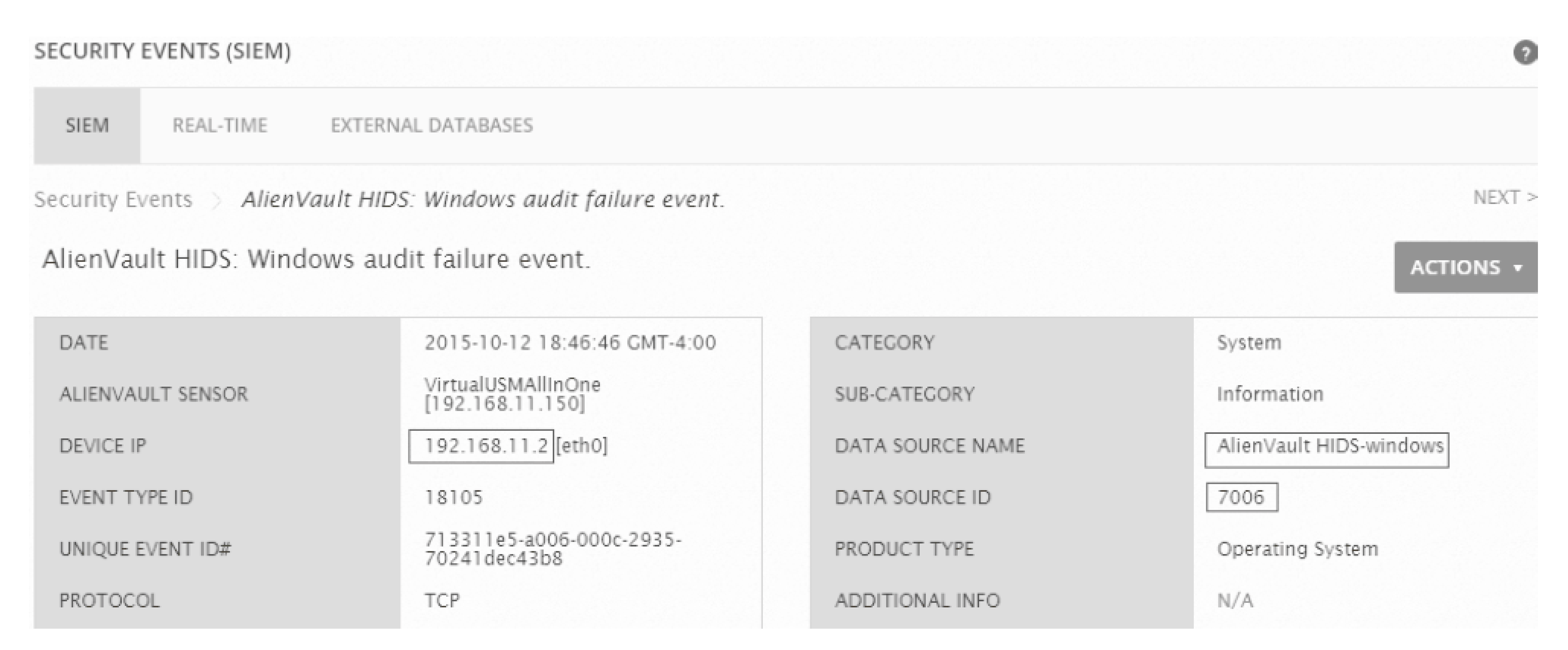


图 9-36　Windows 审计失败事件

这条审计失败的原始日志，由图 9-37 所示。

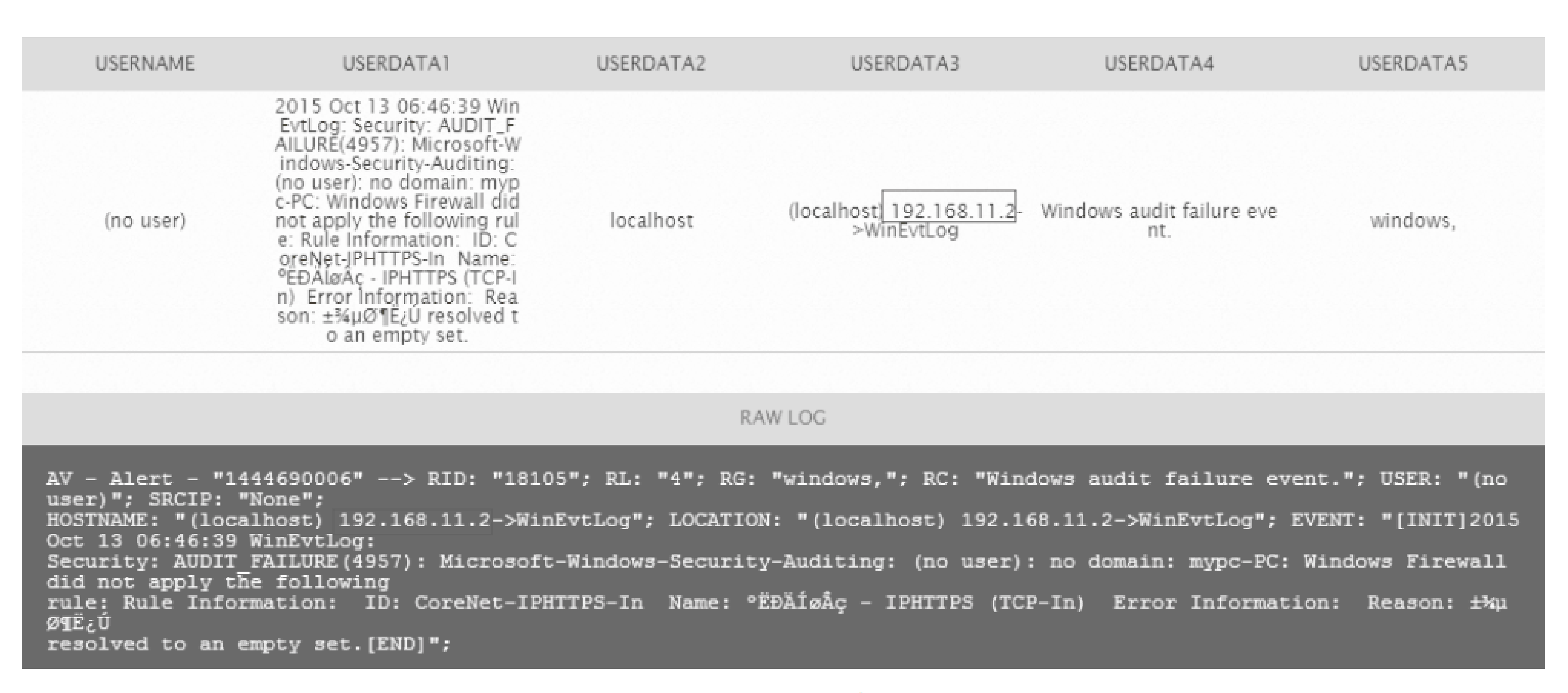


图 9-37　Windows 审计失败事件的原始日志

注：RID 表示规则 IO，RL 表示规则级别，RG 表示规则组。

（4）Agent 快速安装

为方便安装 Agent，系统可以无须到客户机上手动安装 Agent，直接在 Web UI 上完成安装，快速部署 OSSEC Agent 条件有些苛刻：

- 目标计算机为 Windows 2000 Pro 及以上系统，启用 Microsoft 网络客户端和 Microsoft 网络的文件和打印共享服务。
- 调试期间关闭防火墙。
- 用户有 administrator 权限在目标计算机系统安装软件。
- 对于非 Windows 域成员的用户，需要关闭 UAC（User Account Control）

下面看个操作实例：在 Web UI 的 Environment→Detection→HIDS→Agents→Agent Control 中，首先单击“Add Agent”按钮，输入客户机的操作系统名称和 IP 地址。接着单击第 8 个按钮快速部署 Agent 按钮（仅针对 Windows 系统），如图 9-38 所示。

图 9-38　批量部署

接着输入用户名和登录密码，如图 9-39 所示。

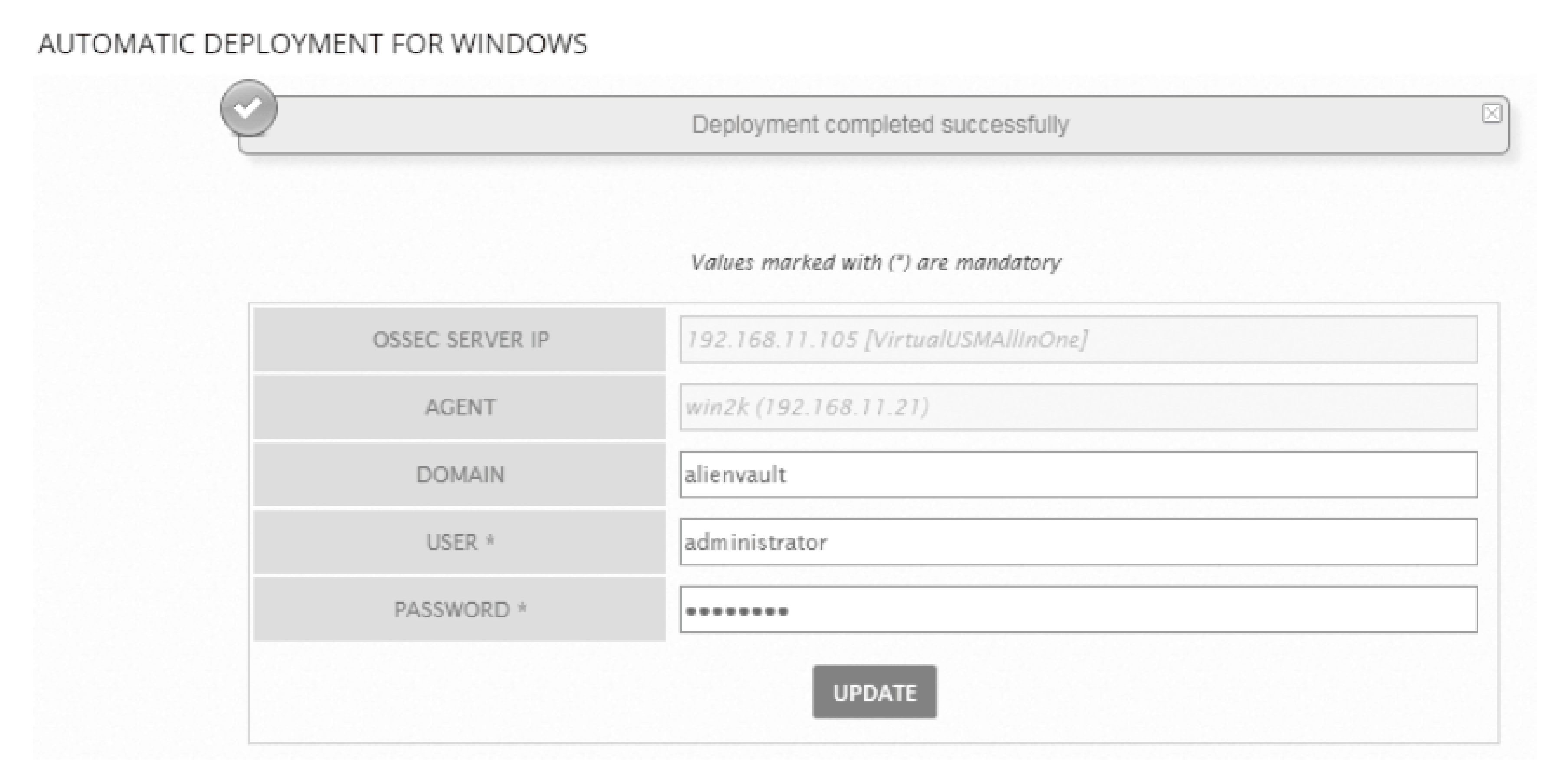


图 9-39　填写验证信息

填写完毕之后单击“Update”按钮安装，即可后台完成，本次实验中 Windows 系统所在 Domain 为 alienvault。

在首次进入 OSSIM Web UI 界面打开设置向导中，系统会逐步引导用户进行 HIDS 的批量部署，但对于实际企业中各种复杂的 Windows 系统而言，不建议盲目采用自动化部署，那样你可能会被各种奇怪的问题所阻碍，所以手动安装或许是更加稳妥的方法。

2. Linux 下安装配置 Agent

安装 AlienVault HIDS Linux 客户端比 Windows 系统更复杂，且不能批量远程部署，需要从源码编译安装，在实际环境中建议大家手动安装，一方面可以深入理解 C/S 工作模式，另一方面排查故障更及时。

（1）准备工作

安装前系统需要 C 编译环境和基本内核，包括 libc 库，下面进行如下准备工作。

基于 Debian 系统，例如 Ubuntu：

```
#apt-get install build-essential
```

基于 Redhat 系统，例如 CentOS：

```
#yum groupinstall "Development Tools" -y &&
#yum install kernel-devel -y
```

（2）安装实例

下面在 Debian Linux 主机（192.168.91.144）上安装 Agent，OSSIM（192.168.91.228）：

```
#cd /usr/src
#wget http://www.ossec.net/files/ossec-hids-2.8.2.tar.gz  /* 下载 */
#tar -zxvf ossec-hids-2.8.2.tar.gz                 /* 解压缩包 */
#cd ./ossec-hids-2.8.2
#/install.sh                           /* 安装，默认目录为/var/ossec */
```

Ossec-hids 软件安装过程中分为以下步骤：

- 选择安装语言，输入 en。
- 选择安装类型，输入 agent。
- 选择安装目录，默认/var/ossec:回车。
- 输入 OSSEC Server 的 IP 地址：192.168.91.228。
- 是否希望运行完整性检测，默认[y]:回车。
- 是否希望运行 rootkit 检测，默认[y]:回车。
- 是否启用联动（active respone），默认[y]:回车。

接下来开始正式编译安装，配置完成后出现以下提示：

```
开始 OSSEC HIDS: /var/ossec/bin/ossec-control start
停止 OSSEC HIDS:  /var/ossec/bin/ossec-control stop
配置文件修改地址为/var/ossec/etc/ossec.conf
```

此时 ossec-logcollector 进程会监控下列日志文件：

```
/var/log/message
/var/log/auth.log
/var/log/syslog
/var/log/dpkg.log
/var/log/apache/error.log
/var/log/apache/access.log
```

程序安装完毕，接着要配置 Agent，包括生成 key，导入 key 启动 Agent 等过程。

（3）产生 Agent Key

在 Environment→Detection 中，选择 Agents，还是单击右侧的“Add Agent”按钮，输入代理名称和 Agent 安装的主机 IP（192.168.91.144），接着单击钥匙图标，产生 Key：

```
Agent key information for '001' is:
MDAxIEhvc3QtMTkyLTE2OC05MS0xNDQgMTkyLjE2OC45MS4xNDQgZjJjNjg3Y2Q5NzIxMTVmYjY
2M2QxYmUwM2UwZGMzMWZjYmQ4NmM4MDRmYWE4NWU5NWY2NDQzNDI4ZThkZWQwNA==
```

（4）导入 Agent Key

在 Web UI 中生成 Agent Key 之后，再返回到 192.168.91.144 控制台。并执行以下代理管理程序：

```
#/var/ossec/bin/manage_agents
```

选择“（I)mport key from the server(I)”。输入大写字母 I。命令行出现“Paste it here(or `\q` to quit)”提示后，便可把 Key 粘贴进去，之后显示：

```
Agent information:
   ID:001
   Name:Host-192-168-91-144
   IP Address:192.168.91.144
```

系统提示确认添加代理？选择“y”，下面就要启动 Agent。

（5）启动 Agent

```
#/var/ossec/bin/ossec-control start
```

（6）查看 Agent 日志

在 OSSEC Agent 端（192.168.91.144）查看 Agent 日志。

```
#cat /var/ossec/logs/ossec.log
```

在 OSSEC Server 端（192.168.91.288）查看收集到的 Agent 日志。

```
#cat /var/ossec/logs/alerts/alerts.log
```

虽然 Agent 发送来的日志队列存储位于/var/ossec/queue/，包括 syscheck、rootcheck 等组件

的队列，它们最终记录在 alerts.log 文件。查看所有 Agent 活动状态，输入以下两个命令：

```
#/var/ossec/bin/agent_control -lc
OSSEC HIDS rootcheck_control. List of available agents:
ID: 000, Name: alienvault (server), IP: 127.0.0.1, Active/Local
ID: 001, Name: Host-192-168-91-144, IP: 192.168.91.144, Active
```

仅列出活动代理：

```
#/var/ossec/bin/list_agents  -a
```

（7）查看 Linux 系统的 Agent

刚刚在 Linux 客户端上安装的 Ossec Agent 工作状态，在 Web UI 查看效果如图 9-40 所示。

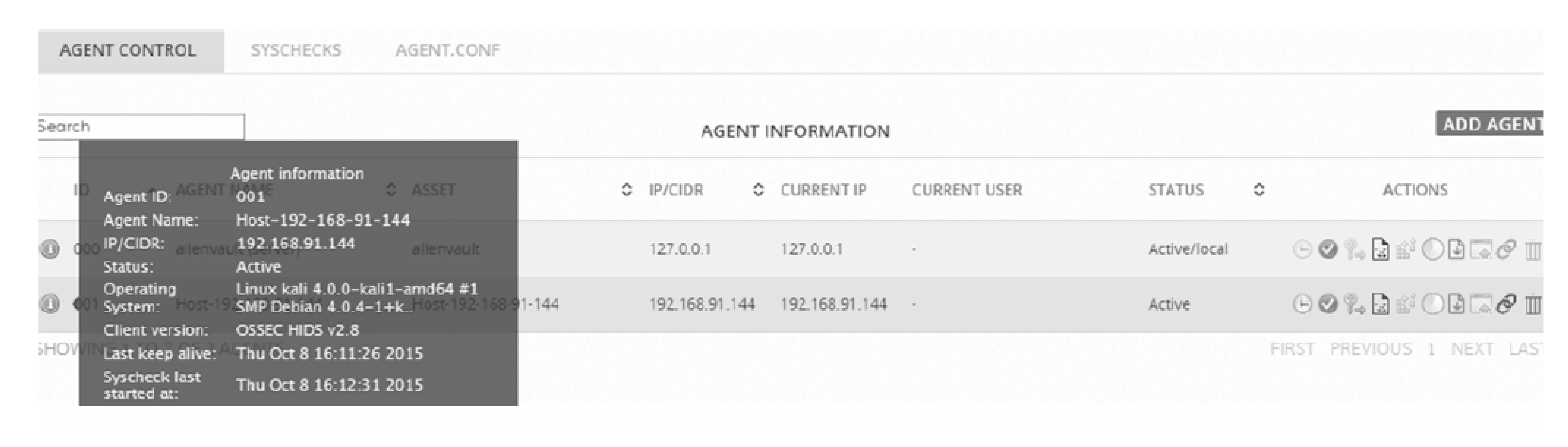


图 9-40　查看 Linux 下的 Ossec Agent 工作状态

（8）检验效果

下面通过在客户机上修改一个文件来检验报警功能。如果我们在客户端 192.168.91.144 上更改 ossec.conf 配置文件：

```
#vi /var/ossec/etc/ossec.conf
```

在该文件中添加一行内容，保存退出，ossec.conf 文件大小发生变化，该文件的 md5 也会发生变化，我们立即可以查看 Ossec 报警。在 SIEM 控制台下，可以看到如图 9-41 所示的事件报告，以及图 9-42 所示的原始日志内容。

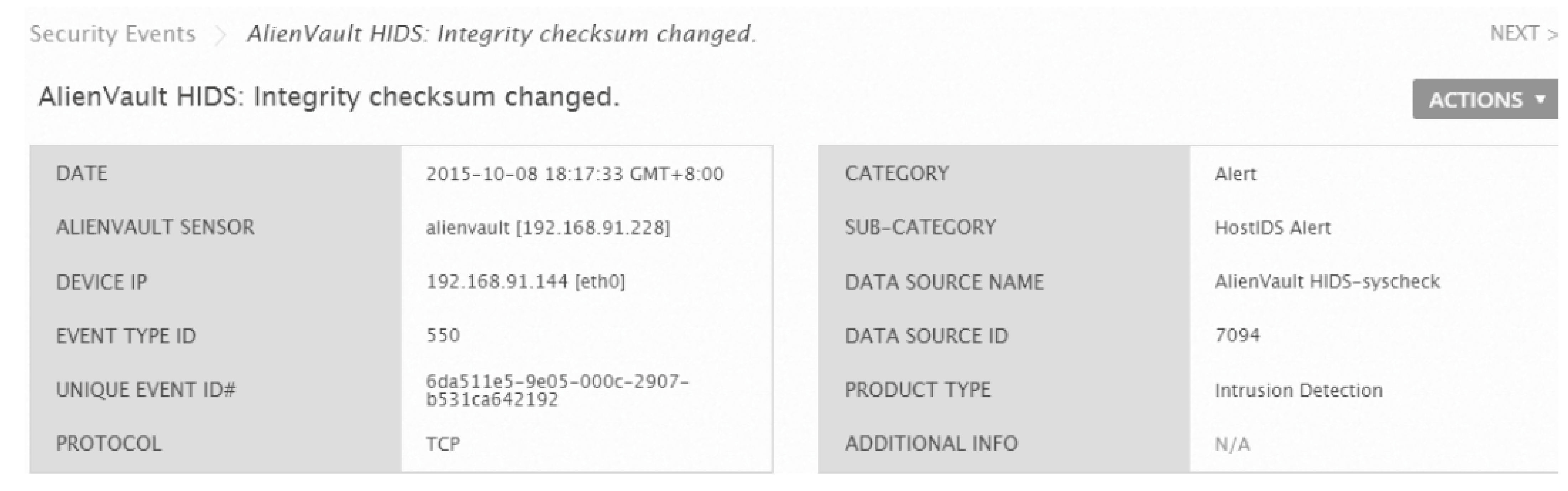


图 9-41　HIDS 发现文件被修改的安全事件

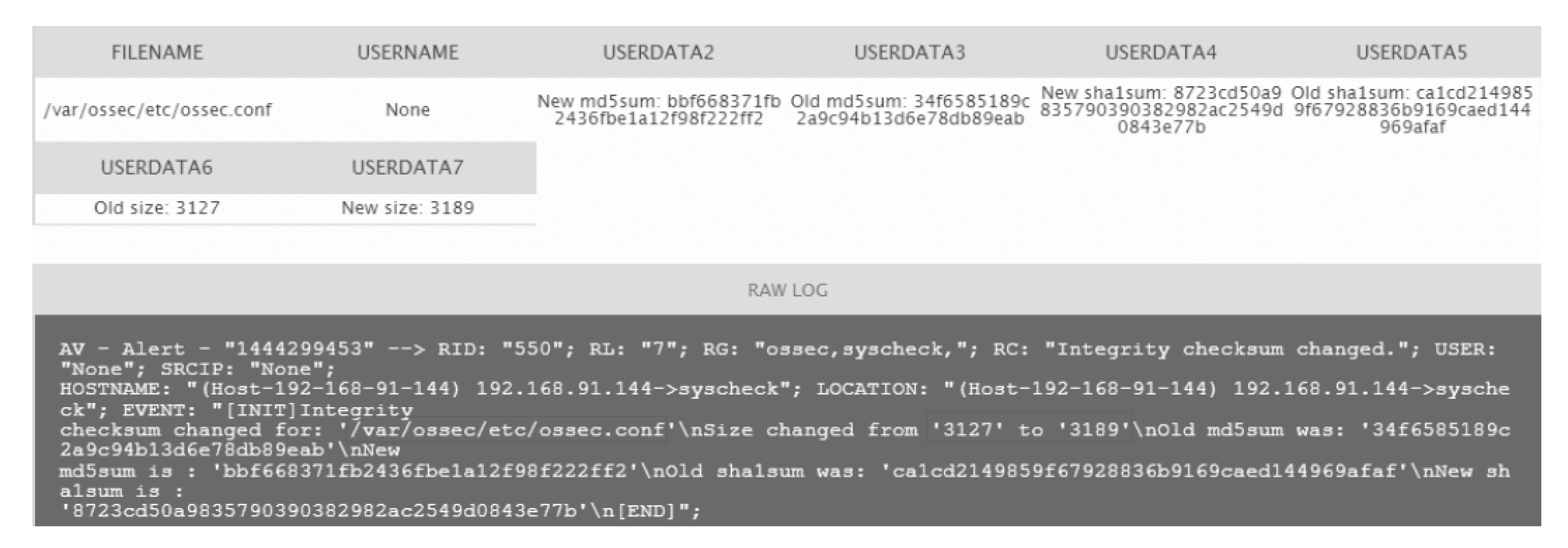


图 9-42　HIDS 发出的原始报警日志

接着下来，我们把以上在 ossec.conf 配置中修改的内容还原，但这次事件还是被 Agent 捕捉并发送到 Server。

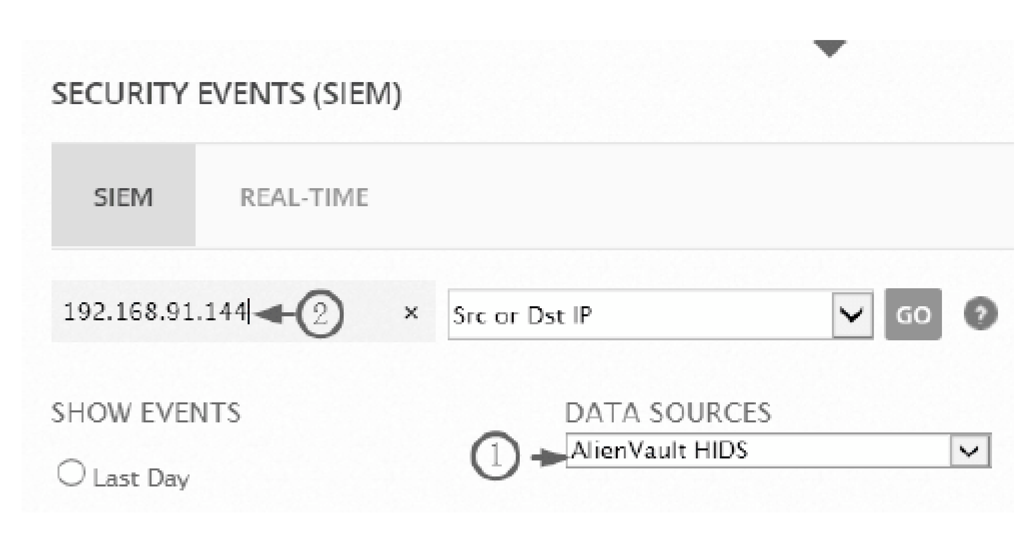


图 9-43　过滤条件

加密日志源源不断地从 Agent 主机发送到 Server，我们要如何快速地分辨各种类型的事件呢？在 SIEM 控制台中，合理利用日志过滤功能可解决该问题。下面以过滤 HIDS 日志为例，首先选择数据源 Data Sources:Alienvault Hids，系统立即显示出查询结果，然后继续输入一个检索条件 Agent 地址 192.168.91.144，如图 9-43 所示。

此时检索出的结果还是非常多，为了将其分类，最后选择 GROUPED 标签，如图 9-44 所示。

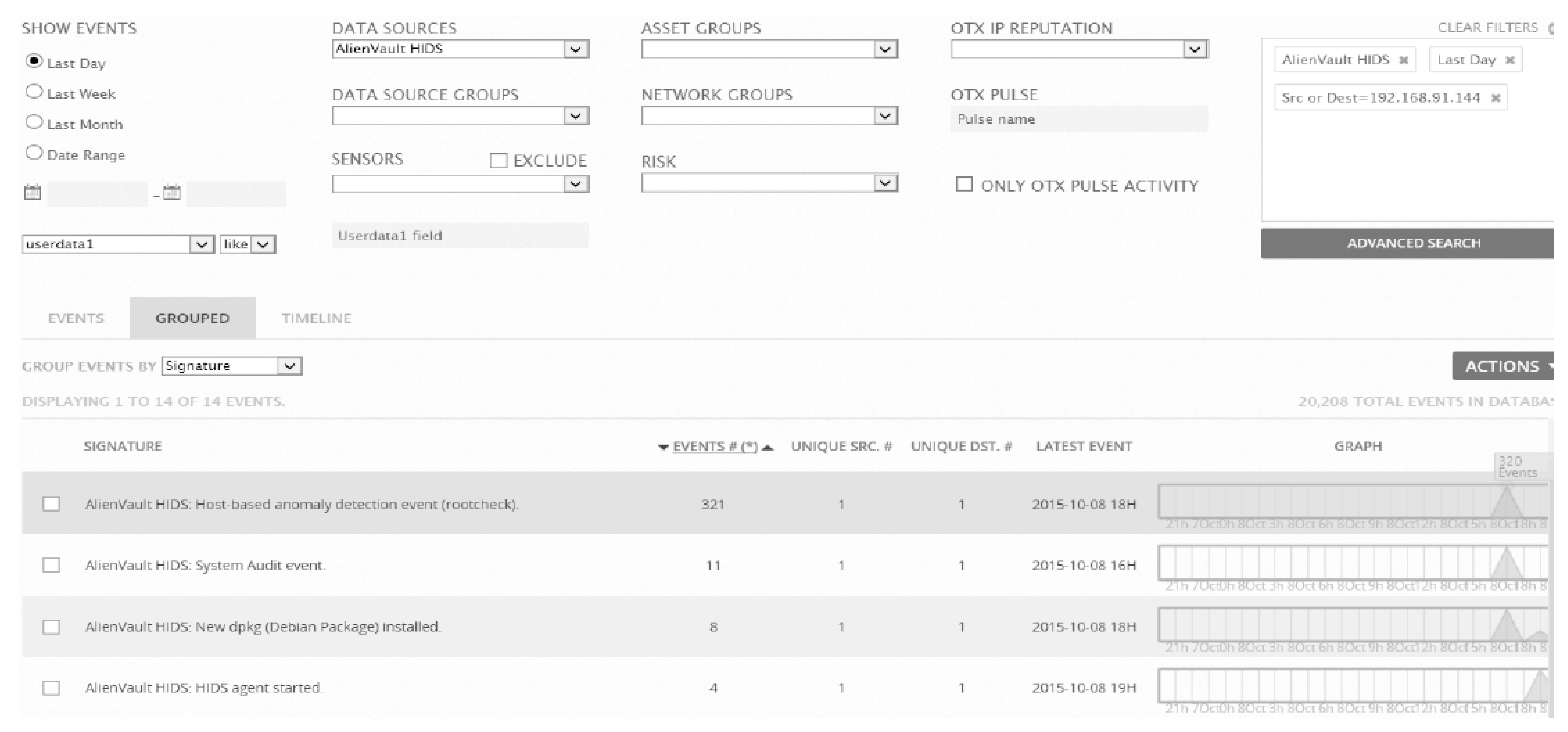


图 9-44　经过多个过滤条件筛选出的聚合 HIDS 事件

3. 典型安装故障举例

在安装 Agent 过程中会发生各种问题，以下列举了常见的 3 个场景：

场景 1：当代理没有启动时，在 OSSIM 上收不到日志，这时我们检查 Windows 系统中 agent 是否启动，在开始→运行中输入“services.msc”命令，打开服务管理控制台，找到 OSSEC HIDS 服务（后台对应 ossec-agent.exe 文件）。

场景 2：通常 OSSEC 服务随系统自动运行，有时服务器端遇到问题，客户机无法连接，可以通过以下命令手工重启服务。

```
#/var/ossec/bin/ossec-control start
#/var/ossec/bin/ossec-control stop
```

场景 3：syscheck 进程停止扫描系统，若让它立即执行，输入以下命令：

```
#/var/ossec/bin/bin/agent_control -r -a
OSSEC HIDS agent_control:Restarting Syscheck/Rootcheck on all
agents.alienvault:/var/ossec/etc
```

以下是关于 agent_control 更多控制选项：

- -h：显示帮助消息。
- -l：列出所有可能的代理。
- -lc：列出活动的代理。
- -i<agent_id>：获取代理的相关信息 agent_id。
- -r：代理中的 integrity/rootcheck 检查，要和-u 或-a 一起使用。
- -a：对所有代理都起作用。
- -u <agent_id> <agent_id>：预先指定代理 ID 号。

启动与停止 OSSEC HIDS：

- 启动：#/var/osssec/bin/ossec-control star
- 停止：#/var/ossec/bin/ossec-control stop

Ossec-server 和 Agent 之间需要用到 UDP 1514 通信，在排除故障过程中，监听该端口显得尤为重要，在 OSSEC Server 端通过 ngrep 命令进行的操作效果如图 9-45 所示。

```
#ngrep  -q -d any port 1514
```

```
VirtualUSMAllInOne:~# ngrep -q -d any port 1514
interface: any
filter: (ip or ip6) and ( port 1514 )

U 192.168.11.2:49836 -> 192.168.11.150:1514
  :+...c..:...U..t.V...L....(@o........j.T.p..m.p.p..\...{.G..kGO...W....M.M....z.6....oo
  .#......V......sy...;$...i.v..K.X..SsS.....k|%&V.}..).......?.M.Y~.V......Z.Q(......La.
  .P.].W`mR!........F.Oc1...$^..Go....G.I....`.X:J...x.&...=.4.U.<r..H.pnD.d\A.......p...
  =...z.\......L.Xp..^+.w...*....RN..l..a...$...Z...3<.\A...R!

U 192.168.11.21:1297 -> 192.168.11.150:1514
  :>]"......`.N.:(.....],.....G9.2*}..E..&H.........>.....#....p....N!b...F

U 192.168.11.2:49836 -> 192.168.11.150:1514
  :+...c..:....l.{a..!.....k&t..Q*.Ow..E.gb...y?4. /..H.O.=.Ows.y$.hE.E..b|..w......*....
  ..F....+....$f....|.q(..pv......\..n......th...G...........=..G..h%6.+"D.......bl...Jco
  )...Ms\.1..\7...+.Bc......w...5..a......;.h.;jk98e......|DR..)P.oWW.....:.......p8ge...
  ..,..z.2.$..i......I......K]I.Q?....;.........lD9F..
```

图 9-45　采用 ngrep 过滤

9.3.9 OSSEC 触发的关联分析报警

当服务器遭受暴力破解时，HIDS 会发送大量失败事件到 Server 端，有关联引擎分析、归纳聚合这些报警信息后，再将告警信息发到 SIEM 控制台，在图 9-46 所示的实例中，展示出一条高风险告警，从告警特征可以明显看出这是针对 Windows2012DC1 主机进行的暴力破解，想要查看详细信息还需要点击这条报警，进入 SIEM 事件继续查看。

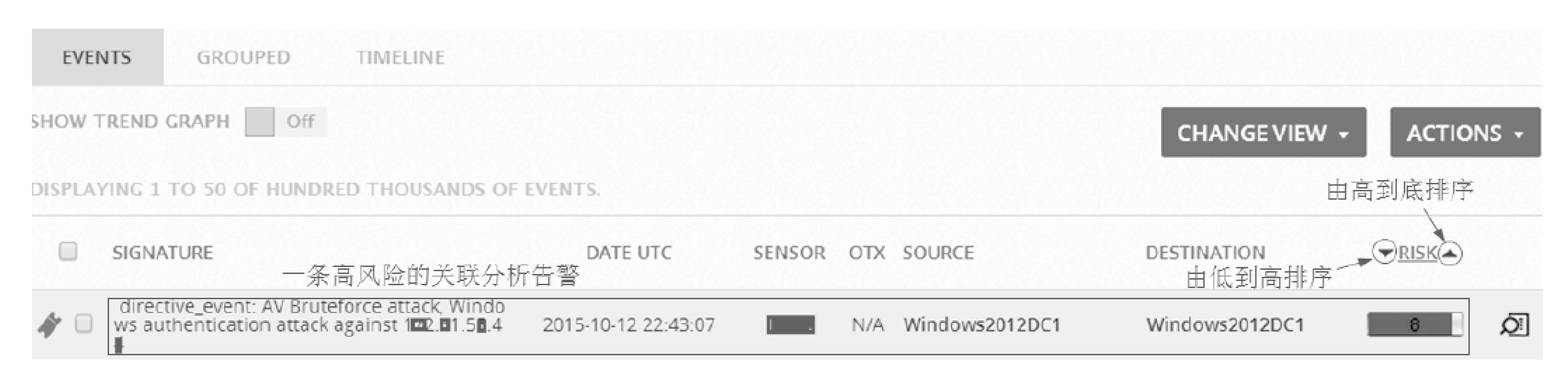


图 9-46　针对 Windows 系统的暴力破解告警

当点击这条事件后，在 Security Events 中立即显示该事件详情以及原始日志，如图 9-47 所示。关联事件名称为“AV Bruteforce attack,Windows authentication attack against 172.x.y.z”。

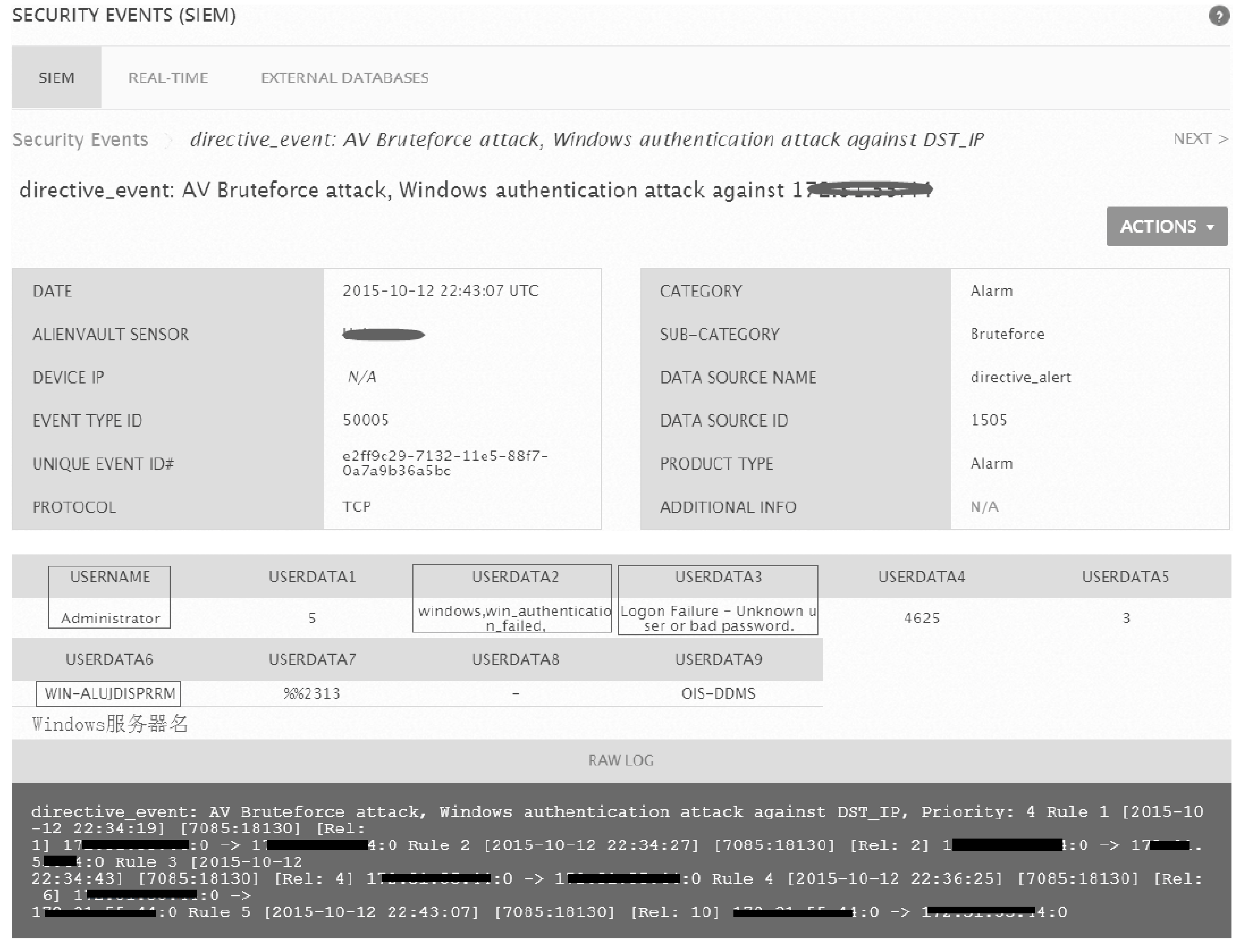
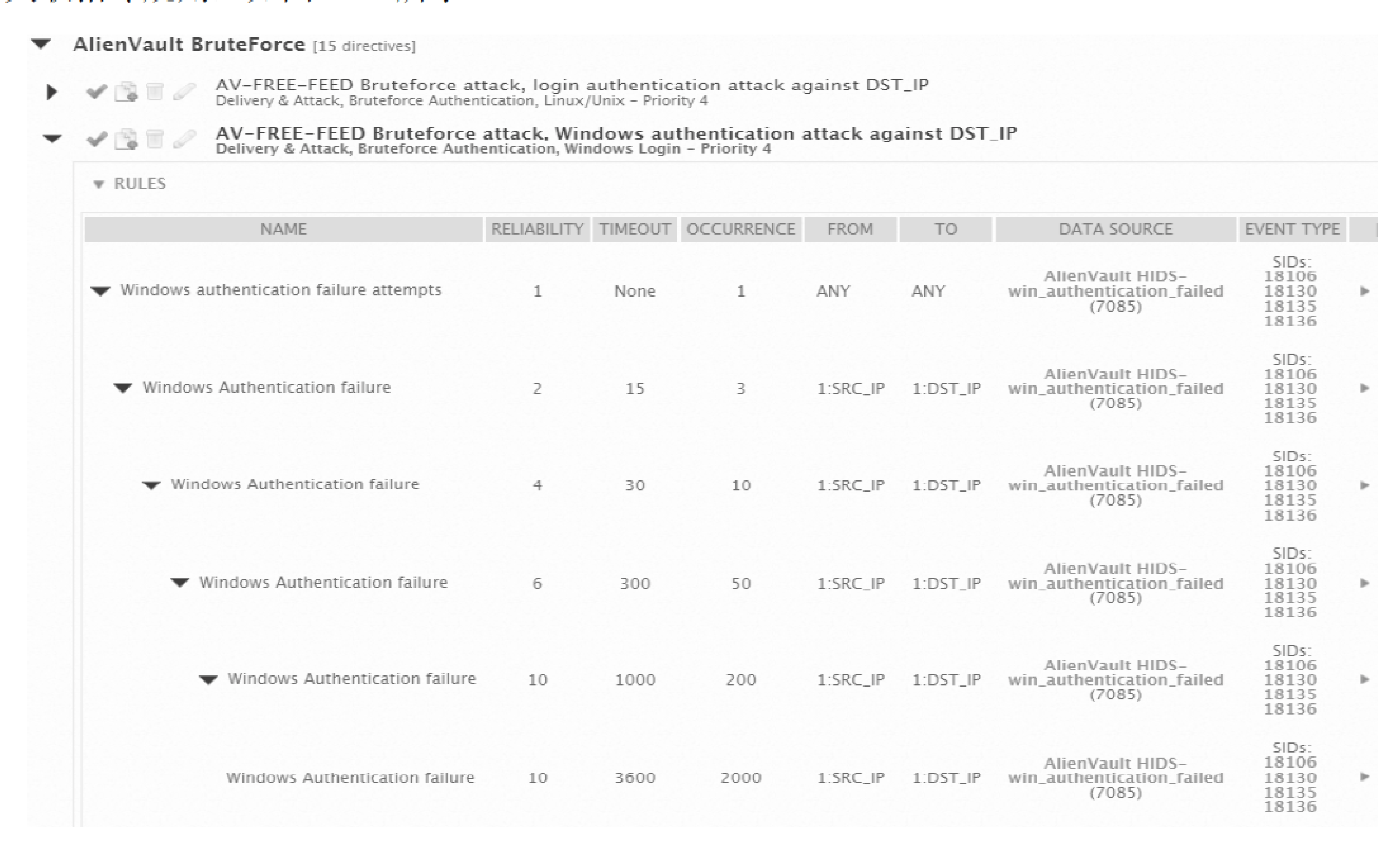


图 9-47　关联分析事件详情

传统 OSSEC 无法将这类关联分析报警展示给用户，而通过 OSSIM 中的关联分析却能实现，关联分析策略在本书第 4 章已经详细讲解过，这里仅介绍触发如图 9-40 所示这条报警的关联指令规则，如图 9-48 所示。



NAME	RELIABILITY	TIMEOUT	OCCURRENCE	FROM	TO	DATA SOURCE	EVENT TYPE
Windows authentication failure attempts	1	None	1	ANY	ANY	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136
Windows Authentication failure	2	15	3	1:SRC_IP	1:DST_IP	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136
Windows Authentication failure	4	30	10	1:SRC_IP	1:DST_IP	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136
Windows Authentication failure	6	300	50	1:SRC_IP	1:DST_IP	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136
Windows Authentication failure	10	1000	200	1:SRC_IP	1:DST_IP	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136
Windows Authentication failure	10	3600	2000	1:SRC_IP	1:DST_IP	AlienVault HIDS-win_authentication_failed (7085)	SIDs: 18106 18130 18135 18136

图 9-48　Windows 暴力破解攻击关联规则

在关联规则中还包含一个知识库，描述这种攻击特征和对策，如图 9-49 所示。

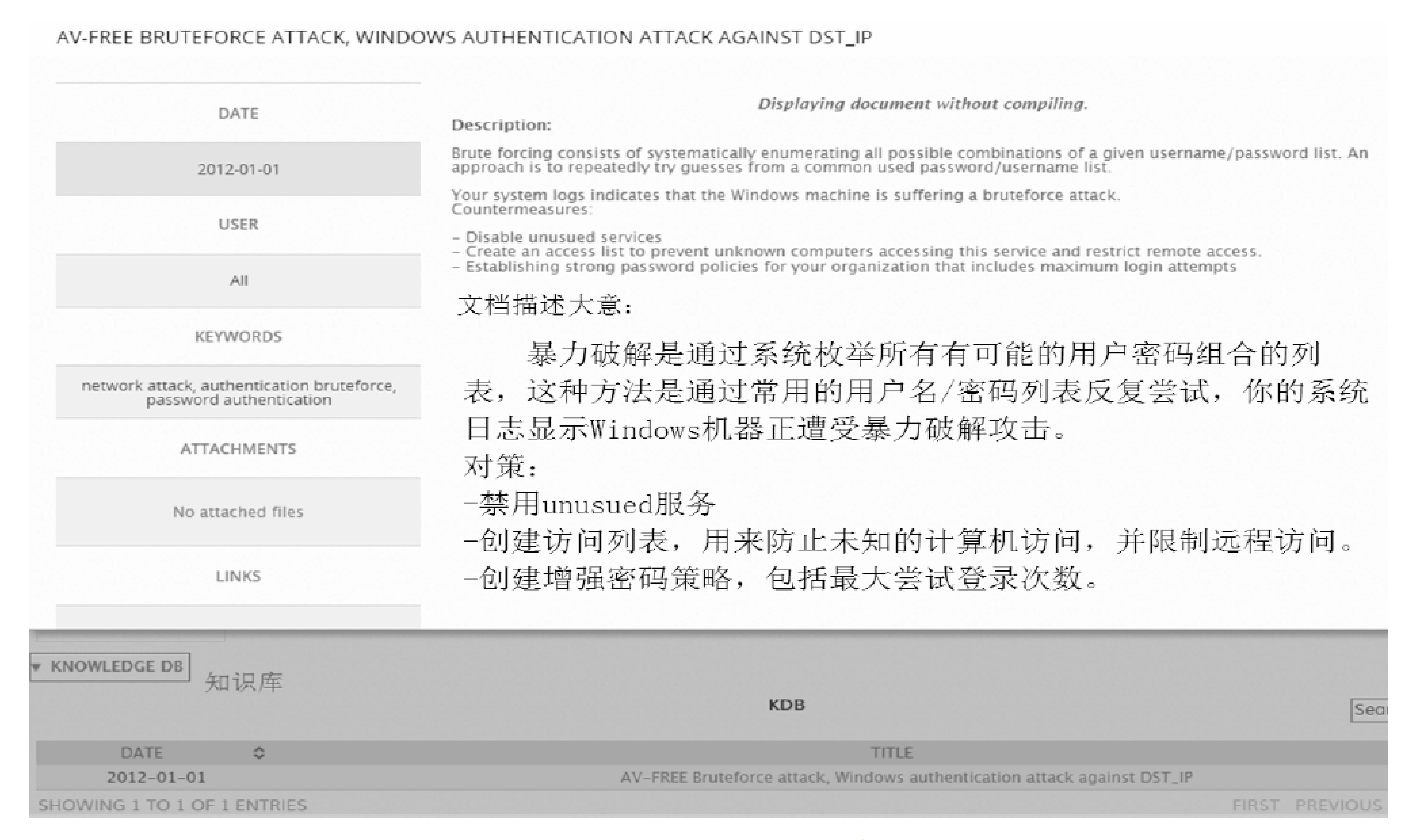


图 9-49　关联分析知识库条目

9.3.10 其他 HIDS 应用

除了以上介绍的文件完整性、Windows 登录失败的报警之外，还有以下几种常见 HIDS 报警需要注意，如图 9-50 所示。在 OSSIM 4.x 版本中所有的 OSSEC Agent 报警以 Ossec 表示，而到了 OSSIM5.0 之后均以 AlienVault HIDS 表示。

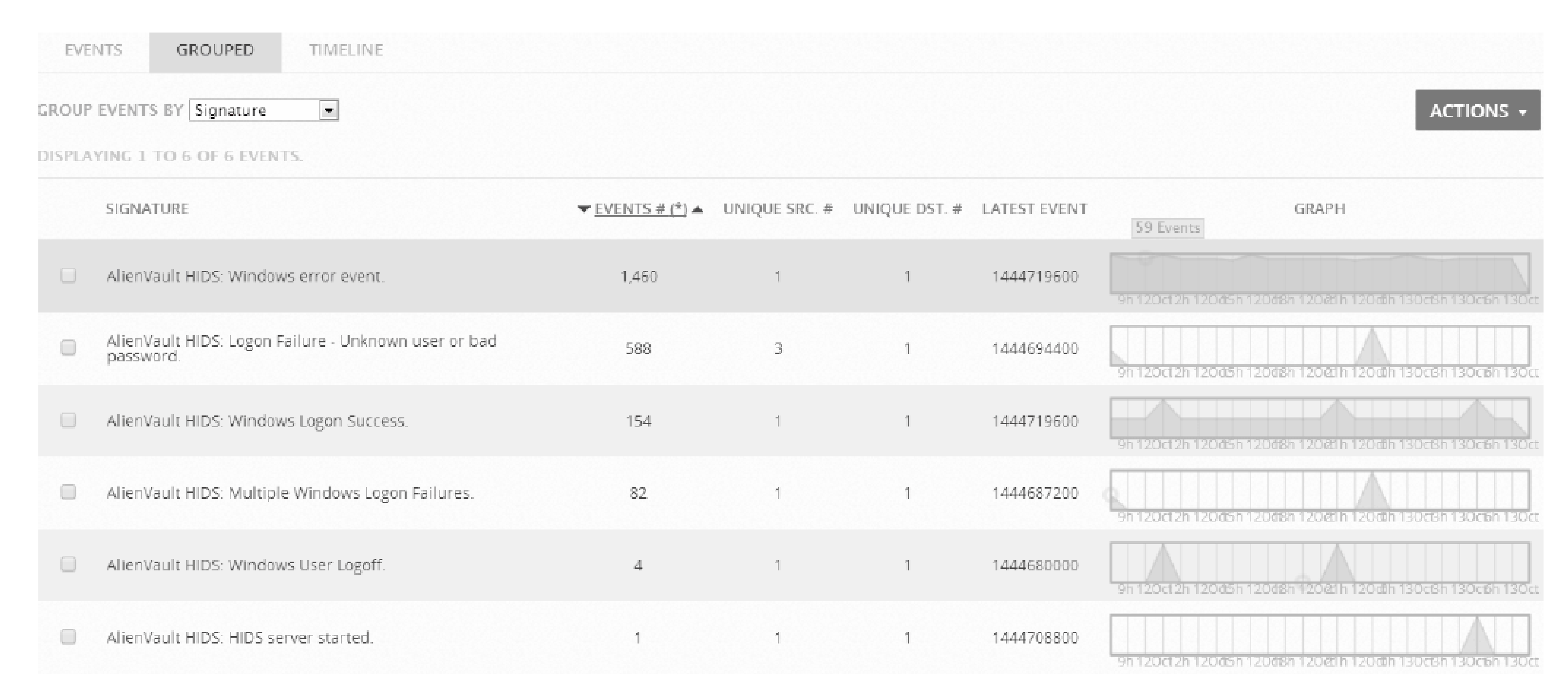


图 9-50 各类 HIDS 报警

此外，还有策略变更报警和时钟修改告警也需要大家注意。

1. 策略更改

只要在 Windows 系统上更改了系统策略，Agent 就会发出报警，在 SIEM 控制台能够看到如图 9-51 所示的告警信息。

图 9-51 策略更改告警

2. 更改系统时钟

一些病毒会在后台修改系统时钟以达到不可告人的目的，服务器系统时钟一旦被修改，会导致非常严重的后果，所以需要密切关注，监控到系统时钟修改的告警如图 9-52 所示。

图 9-52　系统时钟修改报警

9.4 资产 Assets 管理

在 OSSIM 中体现的一种以资产为核心，以安全事件管理为关键流程，基于安全域提供准实时资产风险评估、事件关联、安全预警及应急响应功能的统一安全信息管理平台，所以管理的资产是核心任务之一。在新版 OSSIM 4.8 系统融入了不少资产管理理念，从资产、漏洞和威胁多个维度分析网络风险，从而提升了系统漏洞扫描有效的防御手段。

另一款开源工具 Zabbix，可对资产清单 Inventory 进行管理，可以管理主机 OS 版本、IP、MAC、PORT，但从资产安全运维的角度来看仅了解这些信息还远远不够。资产管理中可展示出资产的漏洞数量、产生网络报警数量、软件包名称、网络服务的状态、NetFlow 数据、监控主机防篡改、对资产的日志分析、标识主机的坐标、在线生成资产报告（合规 ISO PCI 报表、SIEM 事件报表、漏洞评估报告等），而这些重要内容是 OSSIM 最擅长提供的。

9.4.1 资产发现

与 OSSIM 4.1 版本相比，新版 OSSIM 更强调资产的安全管理。对于列出所有资产配置清单的界面在新版体现得不那么重要，而在平台中保存有 IT 资产详细信息，其中包括了资产的 IP 地址，扫描结束后系统根据发现漏洞的 IP 地址，自动将该漏洞归属到相应的资产上。我们利用 OSSIM 中的 Asset Discovery 功能，扫描出监控网段内的所有资产情况，并加入数据库中，如图 9-53 所示。

SCAN RESULTS

☑	HOST	HOSTNAME	FQDN	DEVICE TYPES	MAC	OS	SERVICES
☑	192.168.11.1	Host-192-168-11-1	-	General Purpose	[redacted]	Linux/2.6.X	domain, http
☑	192.168.11.15	Host-192-168-11-15	-	General Purpose	[redacted]	Windows/2000	echo, discard, daytime, qotd, chargen, ftp, smtp, http, msrpc, netbios-ssn, https, microsoft-ds, msrpc
☑	192.168.11.40	Host-192-168-11-40	-	Phone, General Purpose	[redacted]	Windows/Phone	msrpc, netbios-ssn, netbios-ssn, vmware-auth-ssl, vmware-auth, ms-wbt-server, http, vnc-http, vnc
☑	192.168.11.100	Host-192-168-11-100	-	Router, Switch	[redacted]	IOS/12.X	telnet, finger, http
☑	192.168.11.101	Host-192-168-11-101	-	-	[redacted]	unknown	
☑	192.168.11.121	Host-192-168-11-121	-	General Purpose	[redacted]	Windows/XP	msrpc, netbios-ssn, msrpc
☑	192.168.11.221	Host-192-168-11-221	-	General Purpose	[redacted]	Linux/2.6.X	ssh
☑	192.168.11.233	Host-192-168-11-233	-	General Purpose	[redacted]	NetBSD/5.X	netbios-ssn, netbios-ssn, afp, airport-admin, snet-sensor-mgmt
☑	192.168.11.235	Host-192-168-11-235	-	General Purpose	[redacted]	Linux/2.6.X	ssh, http, rpcbind, http-ssl, oracle-tns

图 9-53 发现网络中的资产

对于新发现的资产，我们需要选择“Update database values”按钮添加到数据库，此时注意选择“External Asset”和“Avaliability Monitoring”选项，如图 9-54 所示。

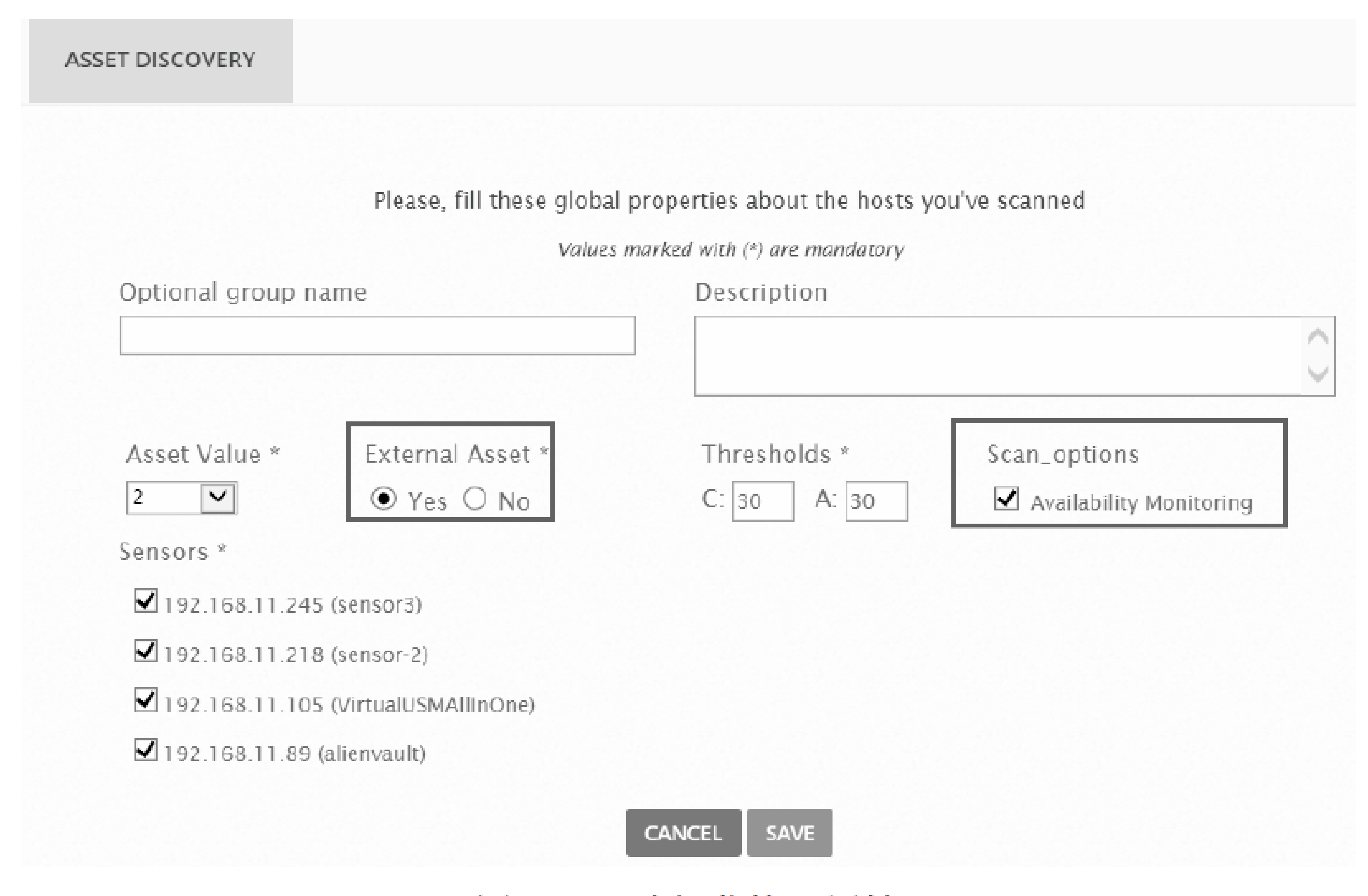


图 9-54 选择监控可用性

在图 9-54 中，如果设置 Optional group name，则可实现将以上资产设置在一个组中。

9.4.2　资产地图定位

接着系统会出现更详细的设置内容，包括主机、IP、FQDN、Location 设备类型等，如图 9-55 所示。图中 Devices Types 设备类型，建议尽量按功能选择。

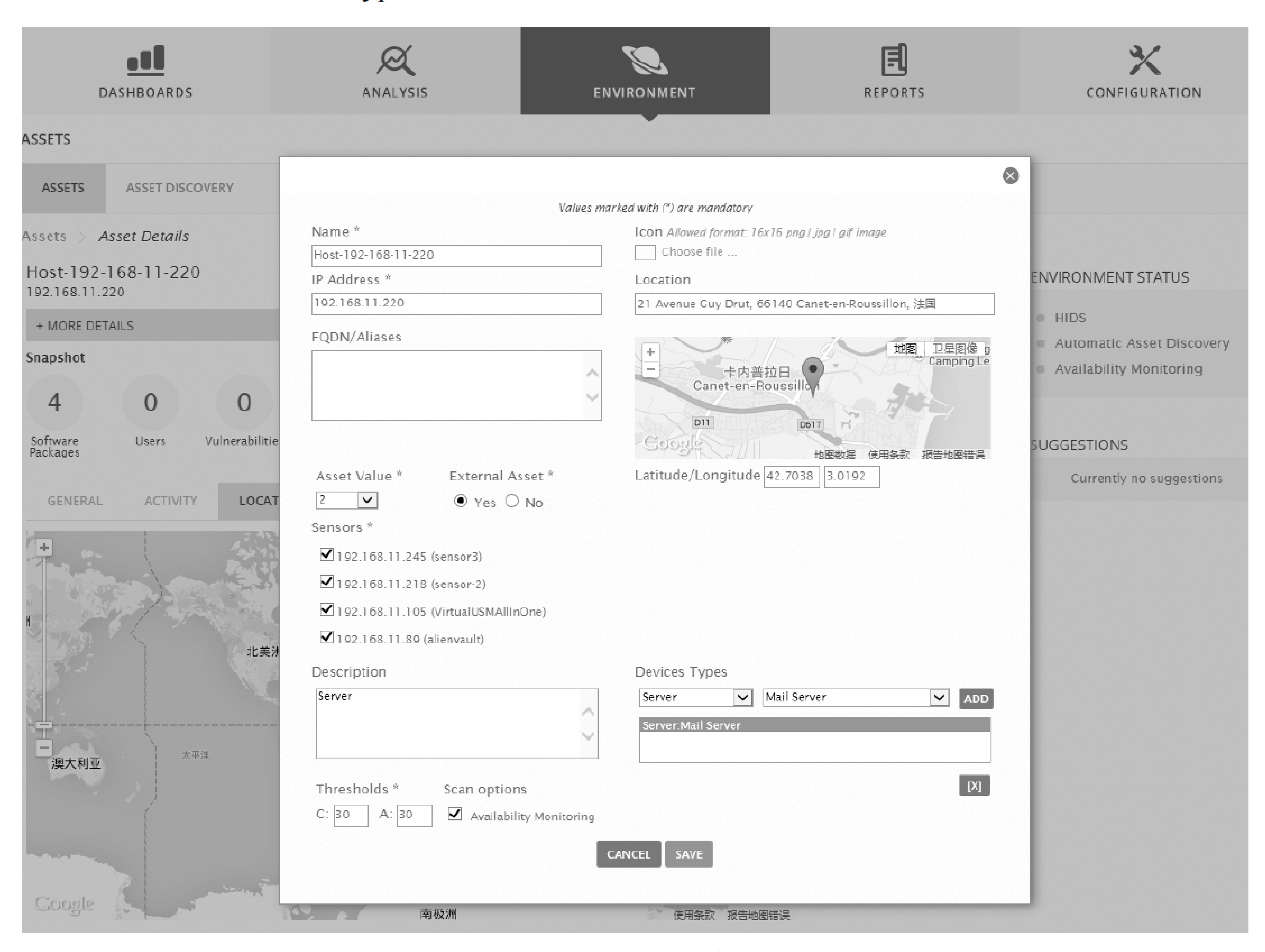


图 9-55　资产定位标示

将扫描结果与资产的安全属性和资产面临的威胁综合分析，即可以呈现出资产的安全风险状况，为风险评估和管理提供依据。通过对业务系统和资产的漏洞情况分析和对比，能为漏洞管理和维护工作提供依据。OSSIM 中资产列表位于 Web UI 中的 Environment→Assets 菜单下，如图 9-56、图 9-57、图 9-58 所示。

9.4.3　扫描控制参数

这个 Web 扫描的选项中共有 6 个级别（0~5），级别越高，扫描速度越快，但也容易被防火墙或 IDS 检测并屏蔽，在网络通信状况良好的情况下推荐使用 T4（即 Normal）。

Timing template 时序模板的选项解释如下：

- Paranoid：作用为了避开 IDS，Nmap 进行所有扫描动作，但频率很低，每隔 5 分钟才发送一个包，所以扫描速度慢。
- Sneaky：和 paranoid 相似，但间隔时间为 15 秒。
- Polite：和 sneaky 相似，只是发送间隔为 0.4 秒，这样不会加大网络负荷。
- Normal：Nmap 默认的预设扫描方式。
- Aggressive：使用进攻性（Aggressive）方式扫描，设定 5 分钟的超时限制，假如同时对多台主机扫描，采用这种方式让 Nmap 对每台主机的扫描时间最多不超过 5 分钟，并且每次探测动作、等待回应的时间在 1.5 秒之内。

而 Insane 与 aggressive 相似，但超时时间为 75 秒，探测回应的等待时间为 0.3 秒，这适用于高速网络环境，其速度最快。

9.4.4 资产列表

当扫描完成之后，即可获得大量成果，这些资产信息会一一列在资产列表中，如图 9-56 所示。稍后我们还需对它们进行识别和矫正（有些信息不一定正确）。

图 9-56　显示资产情况

在资产显示界面中，可一目了然地查看所有管理资产的基本信息，例如主机名称（如图 9-57 所示），收到漏洞数量（如图 9-58 所示），告警数量（如图 9-59 所示），安全事件总数、对资产进行筛选的按钮通过 ADD ASSETS 这三个按钮（分别表示：添加资产、导出资产信息以及一次性删除所有资产列表）实现。

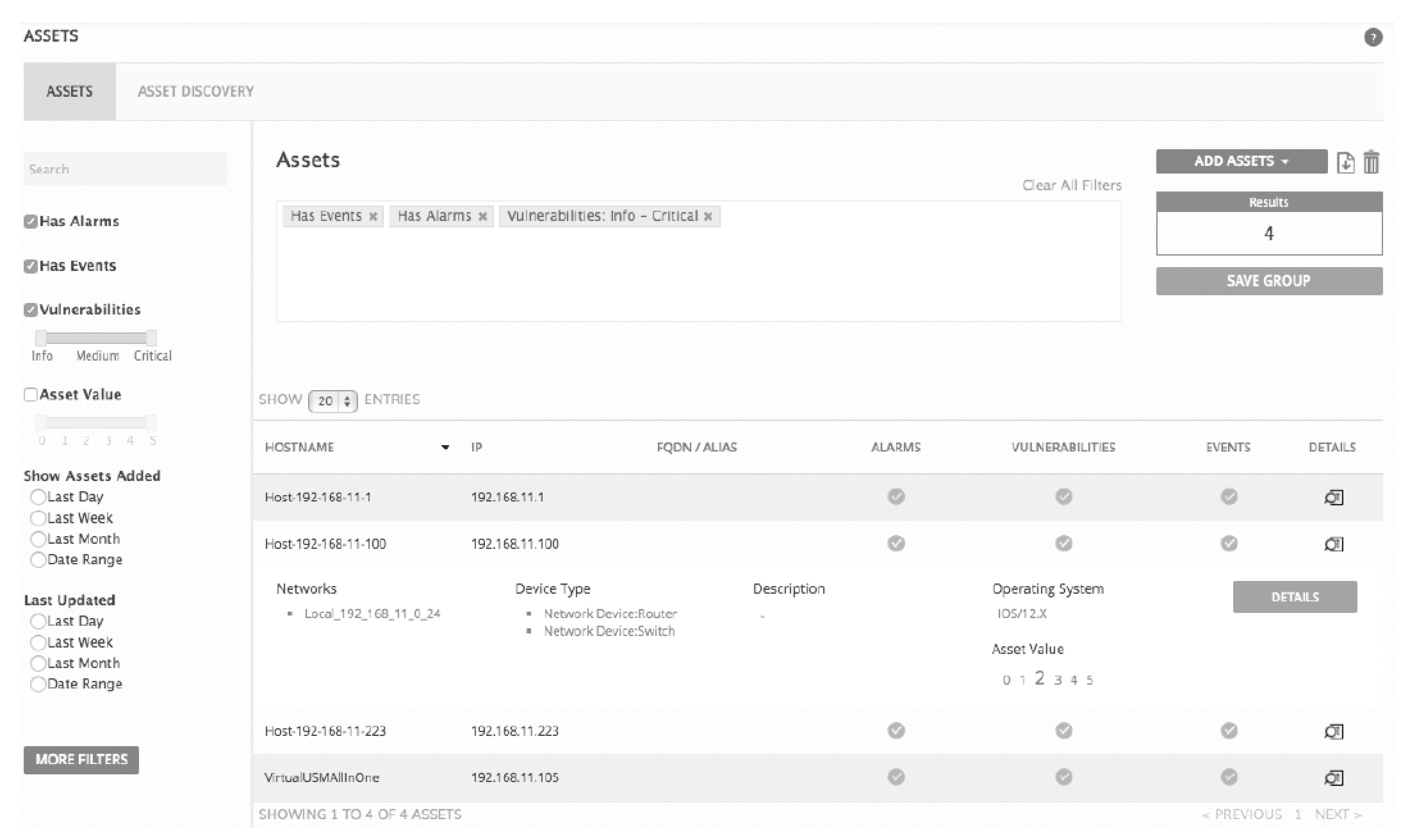


图 9-57　显示资产的事件详情

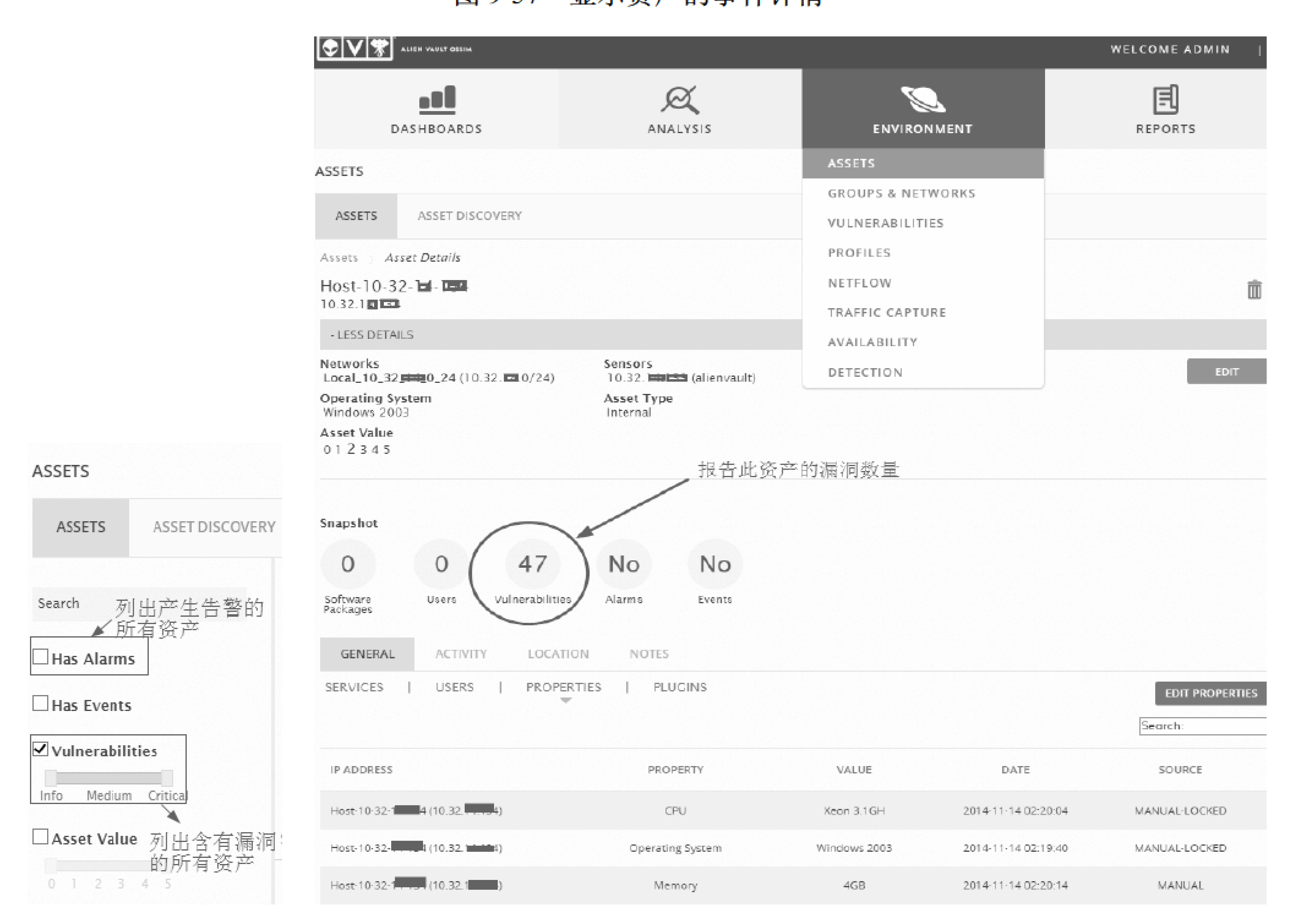


图 9-58　显示资产漏洞情况

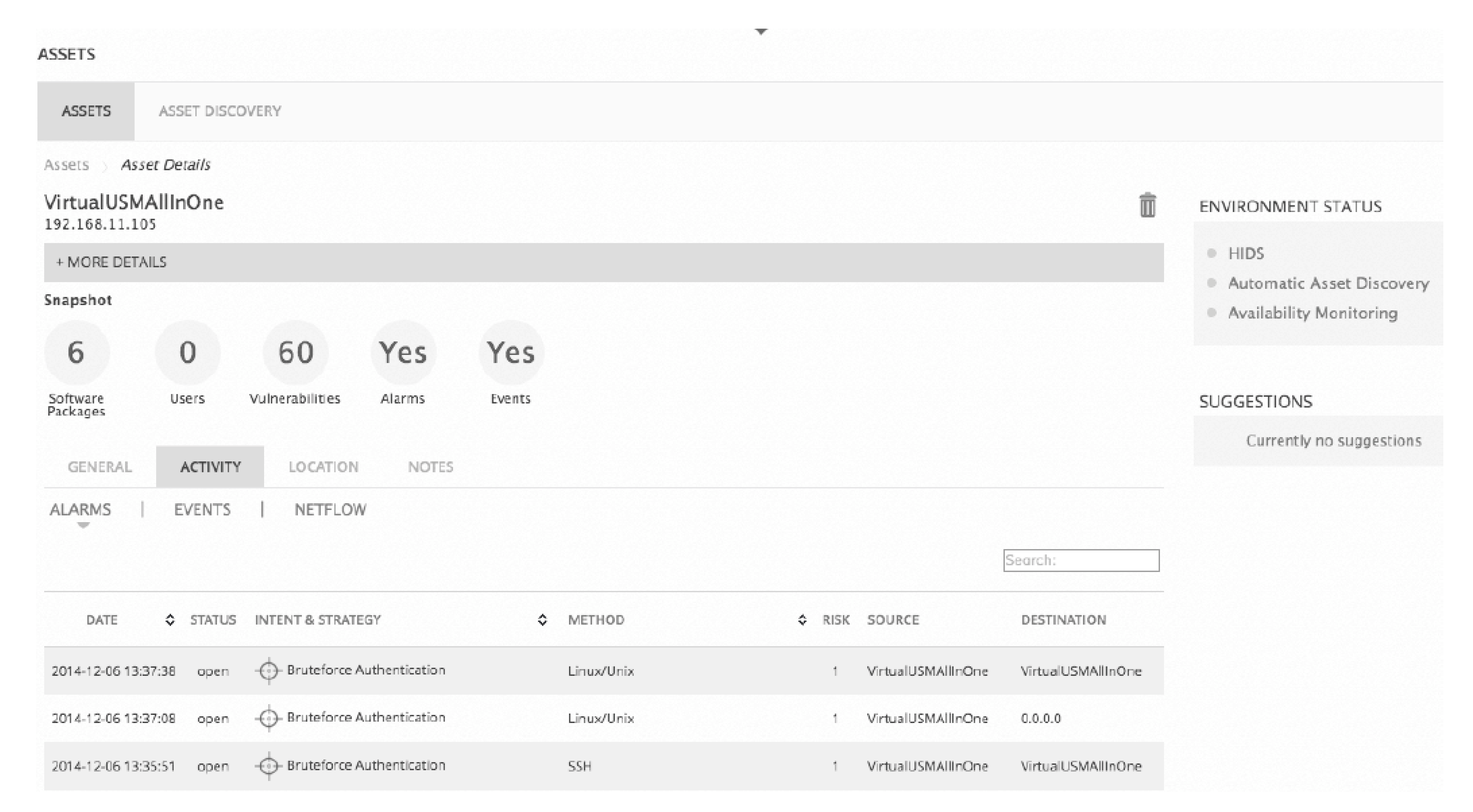


图 9-59　显示事件告警

如果要显示 FQDN/Alias，那么只需在/etc/hosts 文件中，添加主机别名和对应 IP，保存退出后，重新进行主机扫描，在扫描出的结果中更新数据库即可。还记得在 Debian Linux 查看主机名吗？如果忘记了请查看/etc/hostname。

系统中实现资产管理的工具叫做 OCS Inventory-NG（Open Computer and Software Inventory Next Generation），是一款用于帮助管理员跟踪网络中计算机配置与软件安装情况的开源应用程序。在 OSSIM 系统中，利用 OCS Inventory 发现网络上所有的活动设备，例如，交换机、路由器、网络打印机，可以通过 MAC 或者 IP 地址来对它们进行分类。下面我们学习 OCS 的基础知识。

在内网环境中，为了使 OSSIM 快速发现并识别资产，需要将 OSSIM 的 DNS 设置为内网 DNS 服务器地址，而且在/etc/hosts 下要有一张 IP-Hostname 对应表，这样在 SIEM 下的计算机名称可以一目了然。

9.4.5　资产管理工具

OSSIM 下的资产管理工具数据源来自 OCS-NG，由于 OCS-NG 的 Web UI 存在注入漏洞，所以从 OSSIM 4.4 版本开始之后的发行版去除了 OCS 的管理界面，但功能融入 Assets 中。目前版本并不能理想地收集完整的资产信息，还有些信息依然需要用户手动修改。如图 9-60 所示。

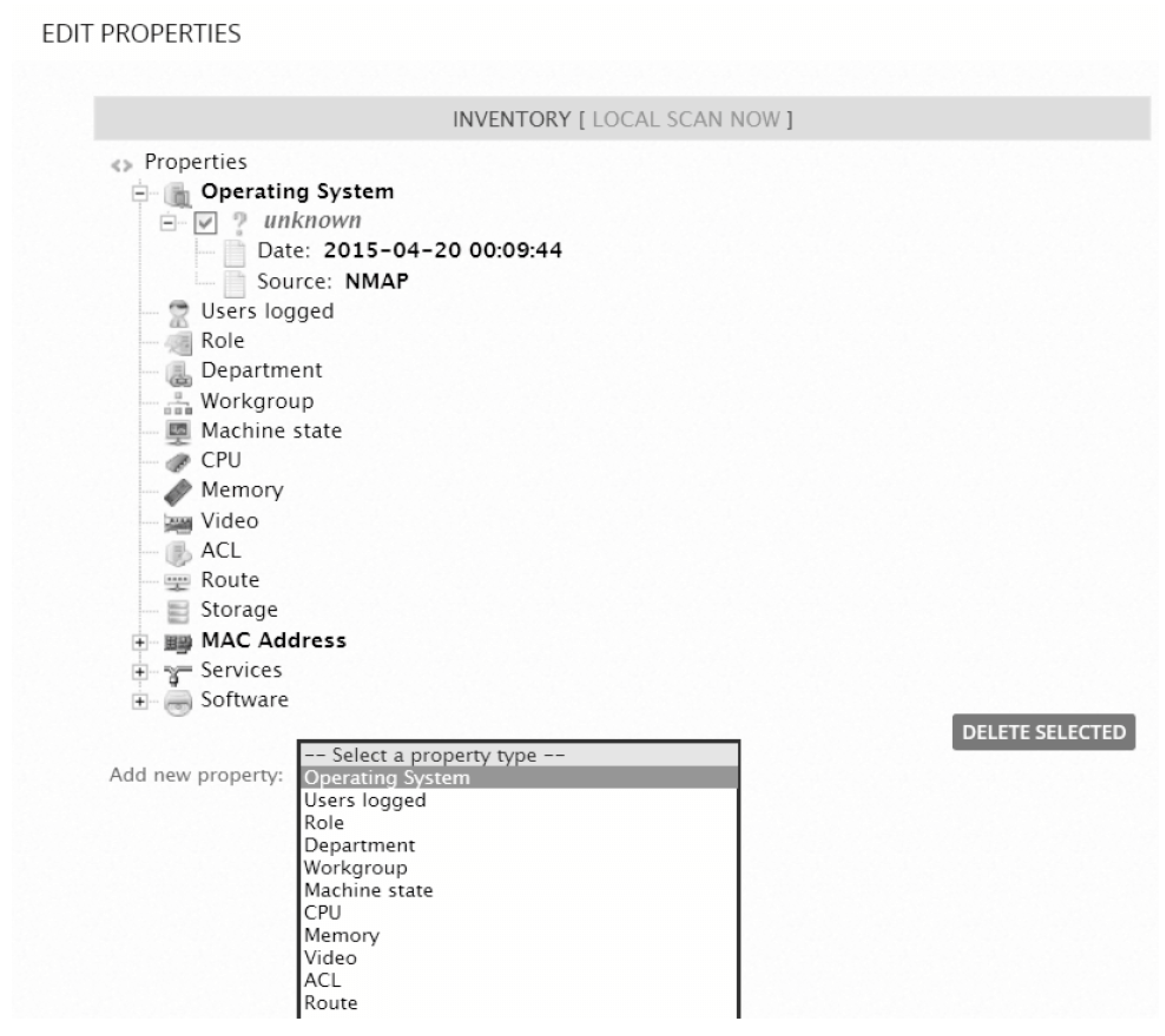


图 9-60　OSSIM 资产信息编辑界面

OCS Inventory NG 通过在客户端上运行一个代理程序（Agent）来收集所有的硬件的信息和软件安装信息。使用管理服务器（Management Server）来集中处理、查看库存清单结果和创建部署包。在管理服务器（Management Server）与代理程序（agent）之间通过 Http/Https 进行通信。代理程序需要安装在客户端计算机上（客户端程序下载位于 OSSIM Web UI 顶部 Support→Downloads 菜单）。

OCS Server 端包括 4 个组件：

- Administration Console：允许管理员通过浏览器来查询数据库服务器的库信息；
- Communication Server：支持数据库服务器与代理之间的 HTTP 通信；
- Data Server：用于储存收集到的客户端的信息；
- Deployment Server：用于储存所有的包部署配置信息。

查看 OCS 数据库，发现很多资产信息存储在 alienvault.asset_filter 表中，该表又与 ocsweb 相关联，我们在 OSSIM 的 Web UI Environment→Asset 菜单中显示资产的内容就来源此数据库。查看 ocweb 库的方法：

```
#ossim-db
mysql> show databases;
mysql> use ocsweb;
mysql> show tables;
```

9.4.6 资产分组

当企业中存在大量服务器和网络设备时，将它们批量导入 OSSIM 系统，必须根据功能或者地域进行分组以便精细化管理和监控。操作步骤如下：

在 Web UI 中选择 ENVIRONMENT→GROUPS&NETWORKS，会进入如图 9-61 所示界面。

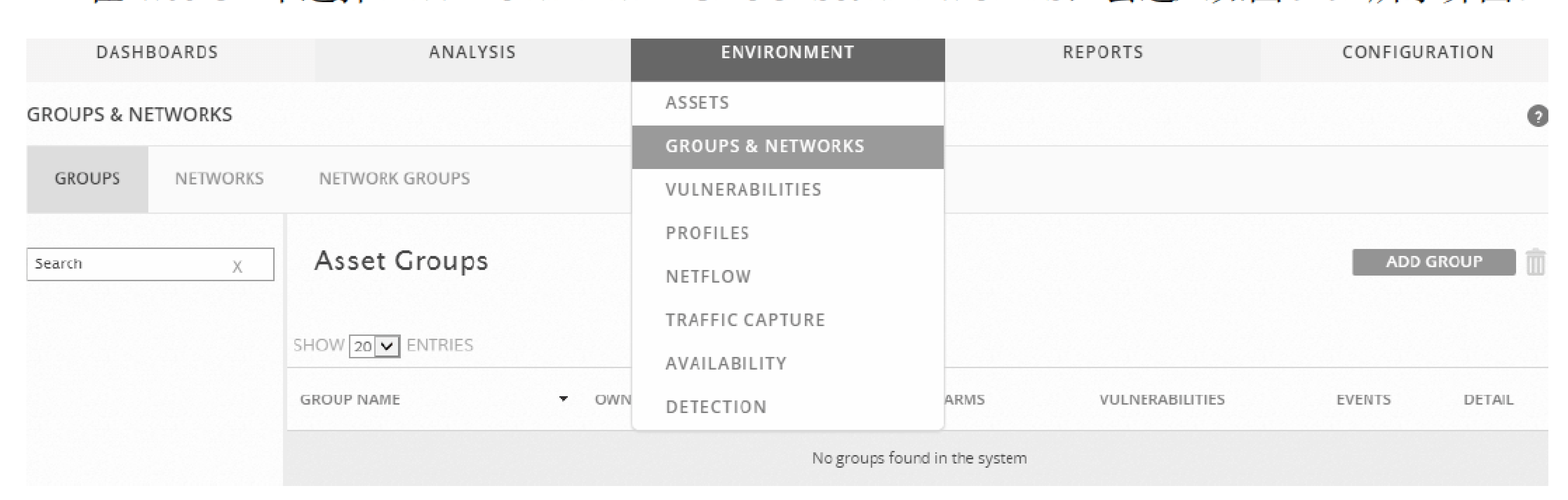


图 9-61　首次添加分组

此时单击图中右侧的 ADD GROUP 按钮添加新的组，系统显示界面如图 9-62 所示。

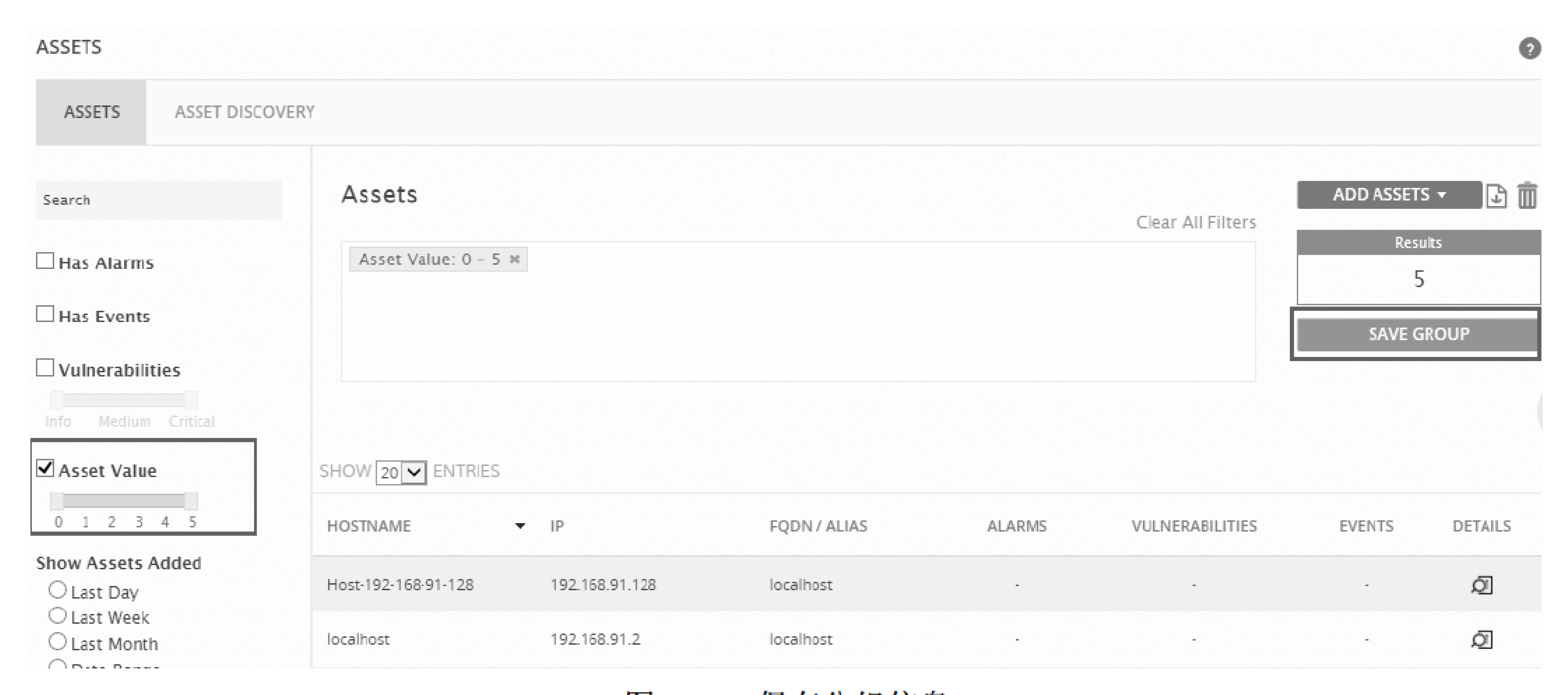


图 9-62　保存分组信息

此时再选择左侧“Asset Value”复选框，根据资产值来快速选取资产，值 0~5 代表所有的资产，通过调节该值大小，可以过滤掉不需要显示的资产，最后单击“保存”按钮，弹出对话框，要求输入“Asset Group Name”组名称以及组的描述信息，再单击“保存”按钮，整个分组过程即完成。效果如图 9-63 所示。

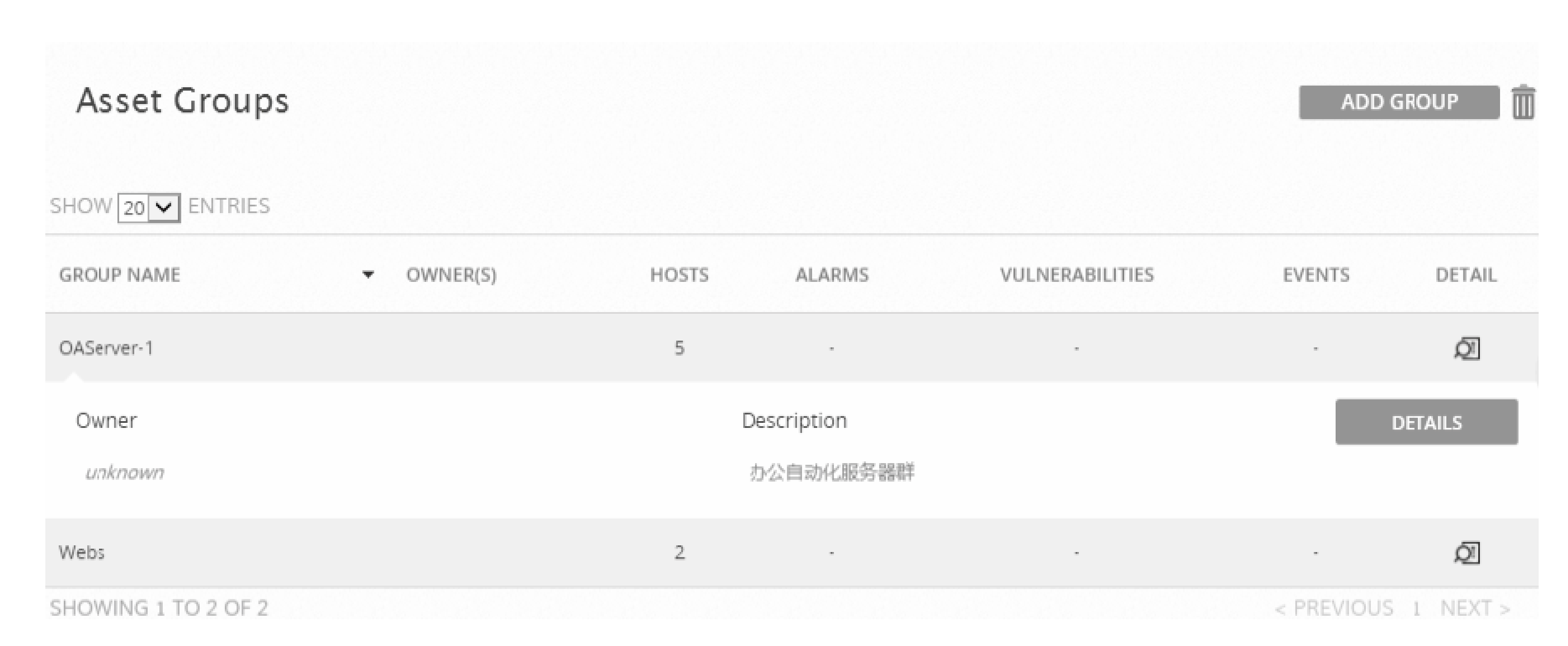


图 9-63　资产分组效果

同样，如果管理复杂的网段，还可根据网络号对网络进行分组。

9.4.7　资产快速查找

当管理成千上万台网络设备时，仅靠分组仍无法快速找到需要的资产，因为显示分辨率的限制。在 Assets 左侧可通过 Alarms、Events、Vulnerabilities、资产值以及资产变化的情况查找，系统提供了更多的筛选条件，例如根据网段、安全软件类型、传感器、设备类型、地理位置以及如图 9-64 所示的服务和端口查找。

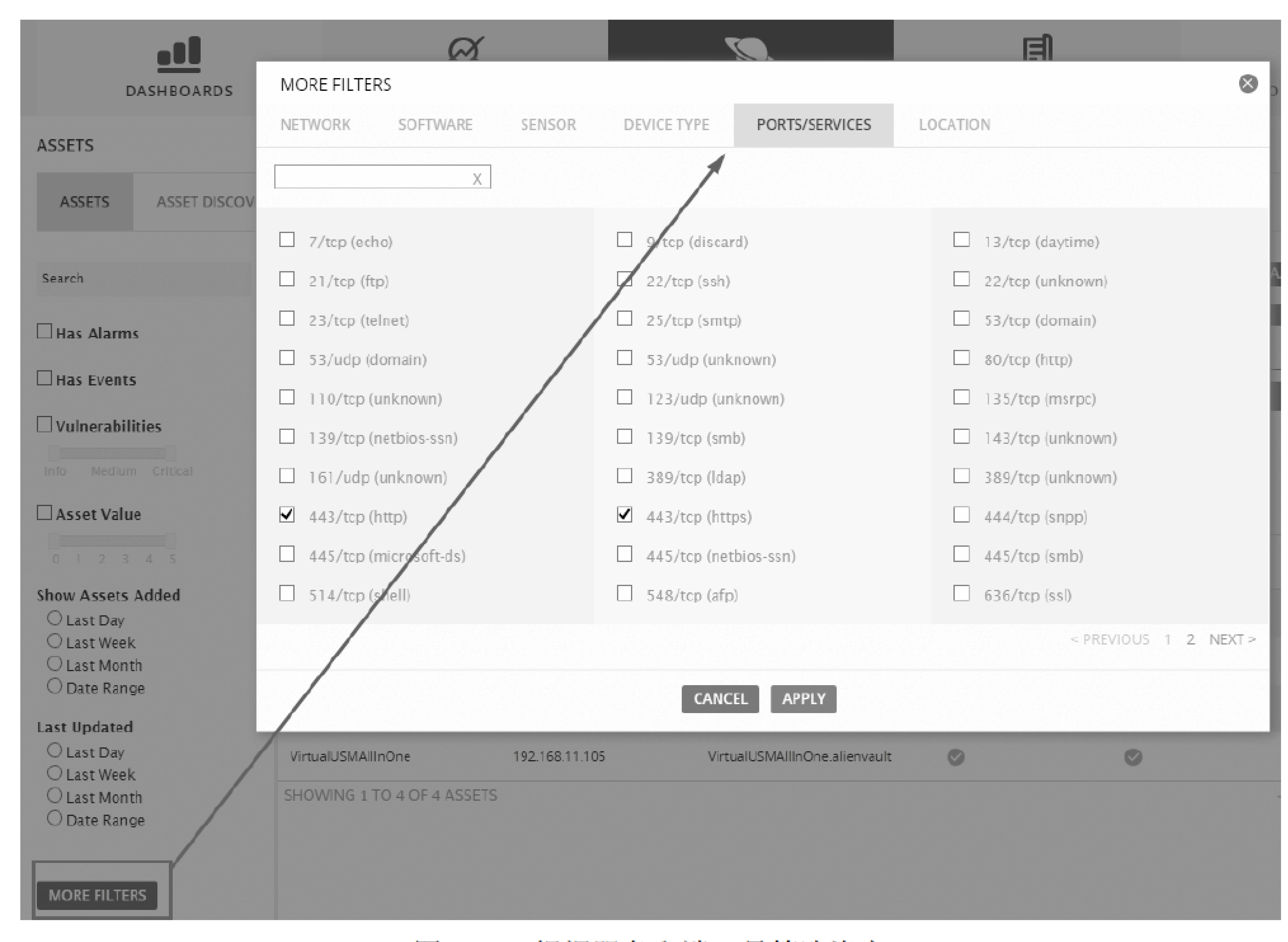


图 9-64　根据服务和端口号筛选资产

9.4.8 设置 Nmap 扫描频率

资产管理时，由于资产总在变化，所以需要周期性扫描以实时发现资产的变化情况。下面开始设置 Nmap 的扫描计划任务，依次选择 Configuration→Deployment→Scheduler，在"NETWORK TO SCAN"一栏填入 CIDR（无类别域间路由）地址，如图 9-65 所示。

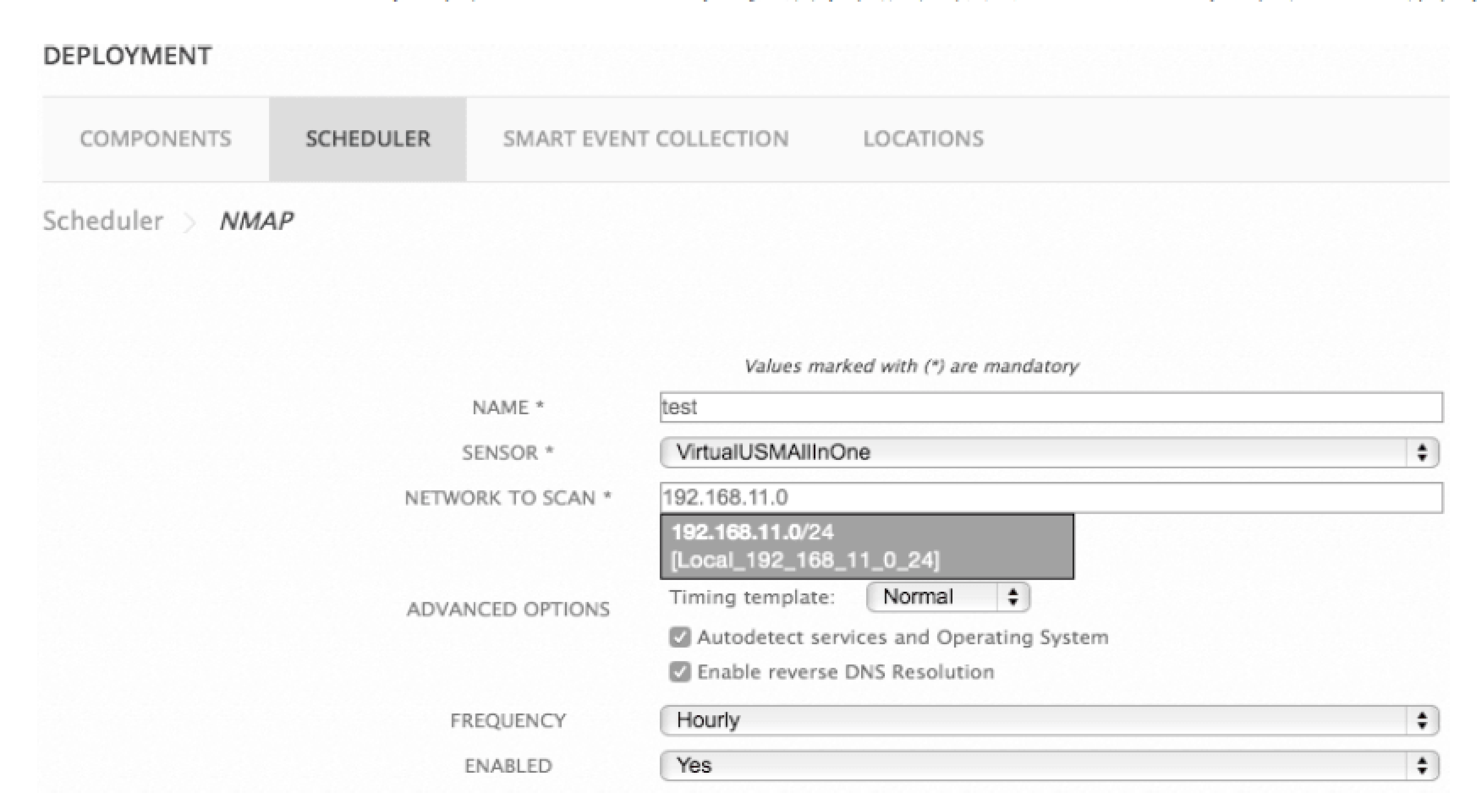


图 9-65 设置资产周期扫描

单击"保存"按钮时，系统提示"The task has been saved successfully"。使用同样的方法我们还可以设置 OCS 和 WMI 的扫描周期。

9.4.9 OCS 检测频率

为了实现 OCS 周期性检测，可在 Configuration→Deployment→Components→Sensors 列表中选择 Sensor，并在显示列表中可发现任务列表，默认 OCS 检测频率为每小时检测一次，如图 9-66 所示。

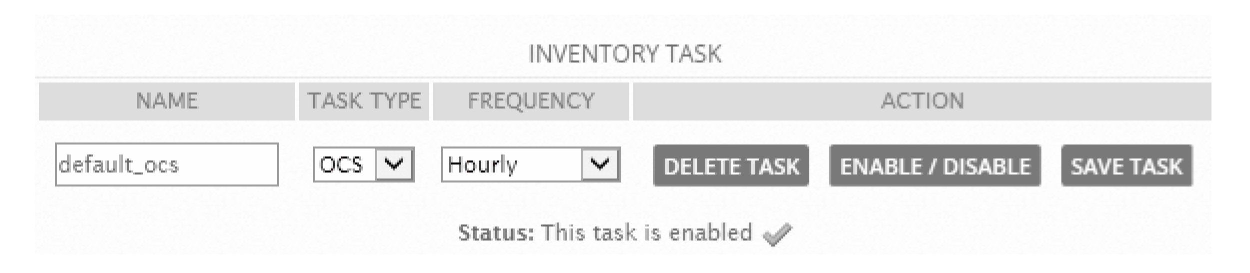


图 9-66 设置频率

9.5 OpenVAS 扫描模块分析

OpenVAS 扫描模块是整个扫描系统的核心，通过对其软件包内约一万多行代码的阅读分析，可以知道该模块包含的两大功能：

- 扫描流程的控制，通过多线程技术实现；
- NASL 脚本的调度与执行，由脚本引擎实现。

9.5.1　扫描流程控制

OpenVAS 可同时对多个主机进行检测与评估，默认情况下是对 5 个主机同时扫描，对每台主机进行扫描时，最大打开 20 个线程，在使用过程中也可根据自己的需要进行修改。配置扫描目标信息时，有多种命名方式，表 9-2 对命名方式进行总结。

表 9-2　OpenVas 检测目标命名方式

命名方式	范例
主机名	Alienvault
IP 地址支持 CIDR 定义 IP 地址范围	单个地址 192.168.11.1 以及 CIDR 表示形式

注意：这里的主机名必须在资产列表中注册，才可正常输入。换而言之，从安全角度考虑，如果新添加的服务器没有注册，系统拒绝扫描漏洞。以下是 OSSIM 下进行 OpenVas 漏洞扫描时，定义检测目标的两种结果，如图 9-67 所示。

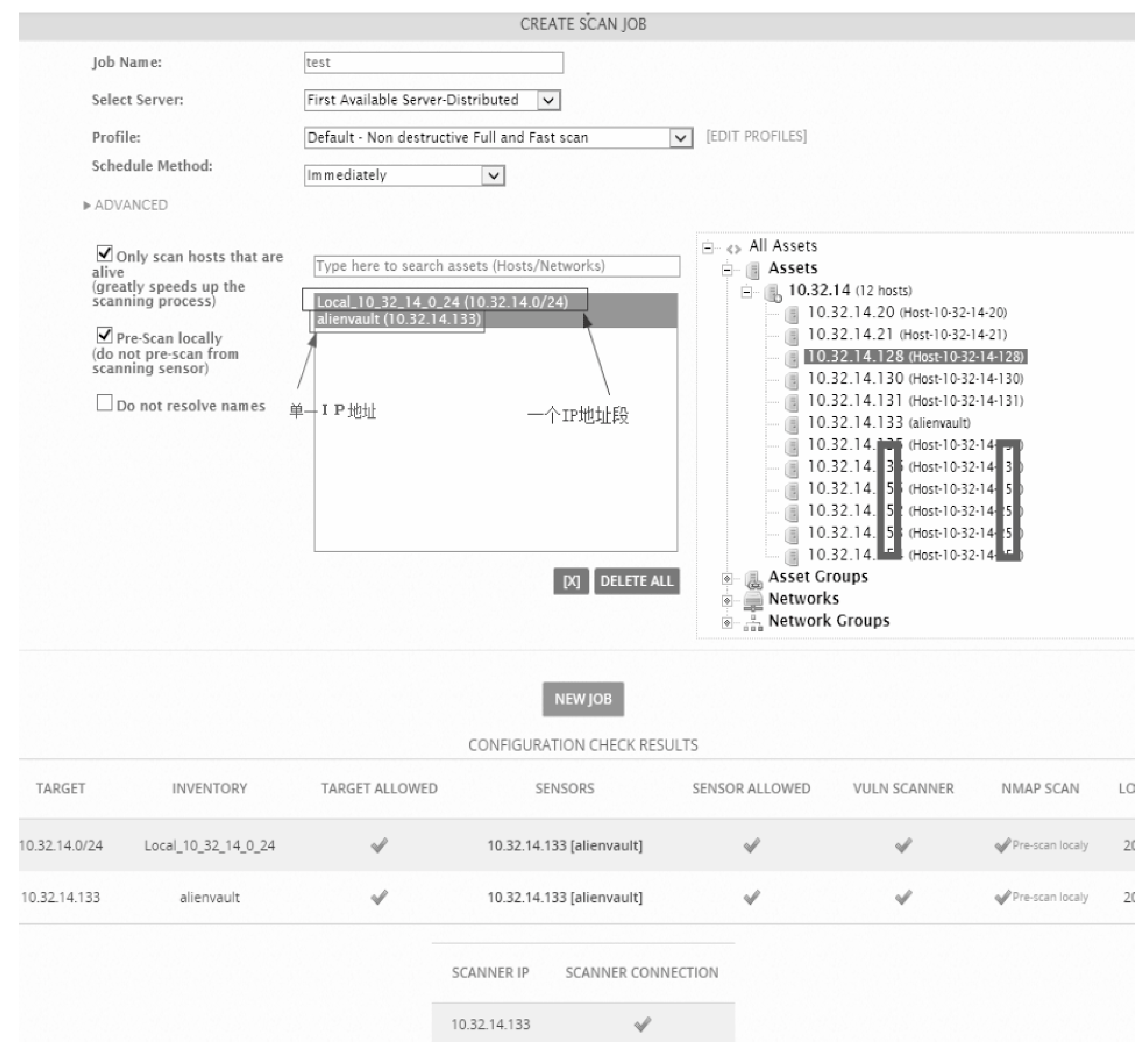


图 9-67　定义扫描主机地址的方法

如果在这里输入检测主机 IP 使用 10.32.11.30~100，或 10.32.11.30~10.32.11.100 这 2 种方式定义目标机器，会提示错误信息。

对扫描目标解析完成后，在不大于同时检测最大主机数条件下，使用 fork()函数对每个目标主机建立相应进程，每个进程中使用多线程调用 NASL 脚本进行扫描。OpenVAS 的扫描流程控制如图 9-68 所示。

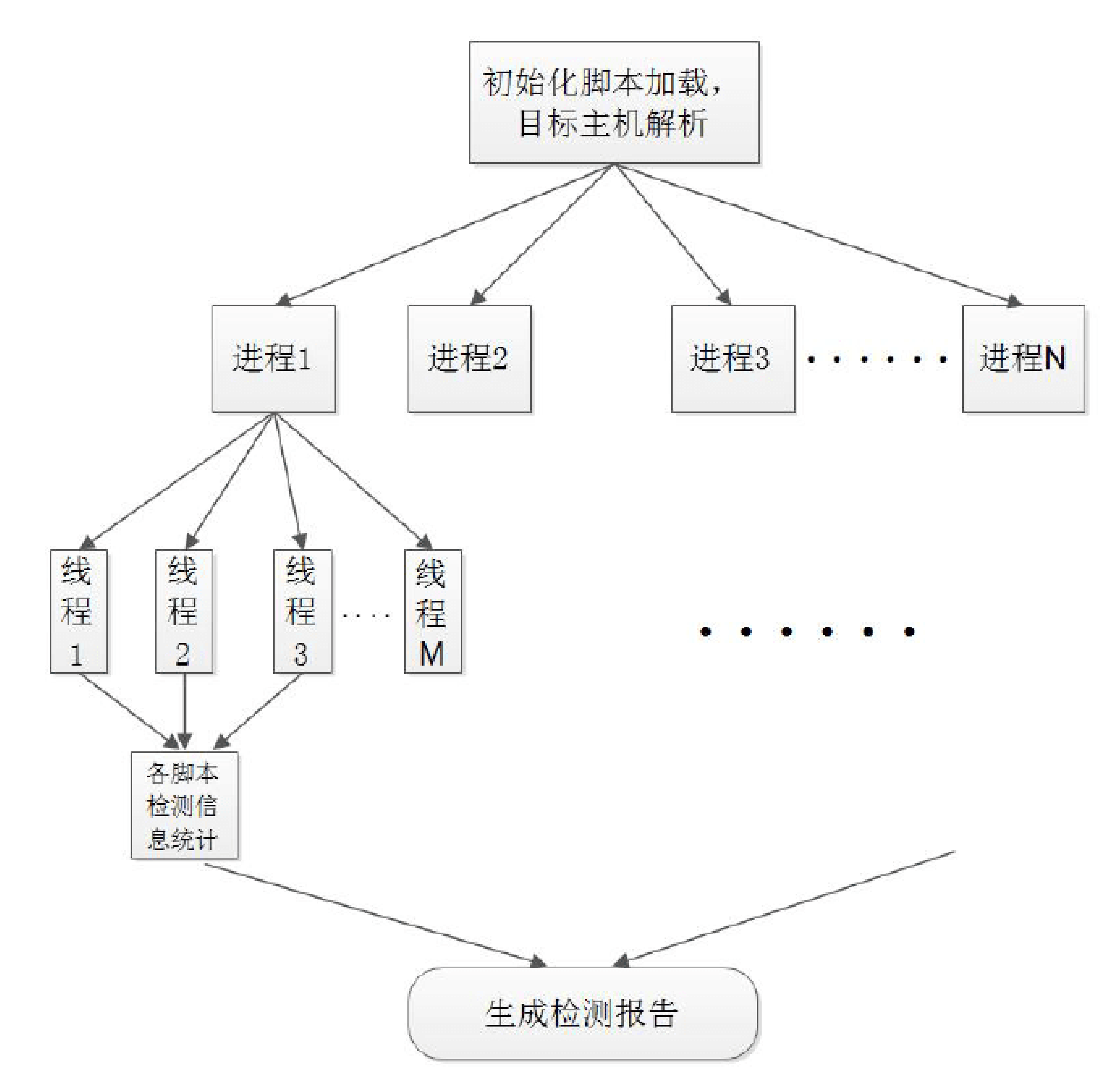


图 9-68　OpenVAS 扫描流程控制

通过多进程与多线程的使用，显著提高了 OpenVAS 检测效率。

9.5.2　扫描插件分析

OpenVAS 采用渗透测试原理，利用扫描器模块中的脚本引擎对目标进行安全检测。Openvas 中扫描器的性能依赖于扫描并发进程数，脚本引擎根据用户提交的配置与要求，首先对脚本进行加载，按照调度顺序依次执行脚本，实现扫描功能。

1. 插件功能举例

在/var/lib/openvas/plugins/目录下，分析 cross_site_scriping.nasl 脚本，跨站脚本漏洞是 Web 应用的常见漏洞。下面将对 cross_site_scriping.nasl 脚本分析，总结此类漏洞检测过程。查看这个脚本第 77 行：

```
77行  script_category (ACT_GATHER_INFO)
```

设置脚本的类别，这些类别有：

- ACT_GATH（信息采集类脚本）
- ACT_ATTACK（尝试获取远程主机权限脚本）
- ACT_DENIAL（拒绝服务攻击脚本）
- ACT_SCANNER（端口扫描脚本）

Category 代表了脚本执行的优先级，这种优先级的设定符合常规扫描过程，优先级如表 9-3 所示，该表在调整扫描策略时会体现出来。在扫描初期需要收集目标主机的端口、系统、服务等信息，根据这些已知的信息进行后续测试，这种测试也是按照由弱到强的顺序执行，为什么呢？如果首先就进行最强的漏洞扫描，相当于进行最危险的测试，一旦目标主机存在相应漏洞，系统立即就会崩溃，整个测试就无法全面进行。

表 9-3　NASL 脚本优先级

Category 类型	优先级
ACT_INT	0
ACT_SCANNER	1
ACT_SETTINGS	2
ACT_GATHER_INFO	3
ACT_ATTACK	4
ACT_MIXED_ATTACK	5
ACT_DESTRUCTIVE_ATTACK	6
ACT_DENIAL	7
ACT_KILL_HOST	8
ACT_FLOOD	9
ACT_END	10
ACT_UNKNOWN	11

NASL 脚本的结构与参数在后面将会进行详细介绍，此处先对脚本中 category 参数进行简单描述。category 在脚本调度的过程中非常重要，category 代表了脚本执行的优先级，通过对各类 category 值的优先级进行定义，NASL 脚本按照表 9-4 中给出 category 优先级由高到低执行，相同优先级按照脚本 id 顺序执行。

这种优先级的设定符合一般扫描过程的逻辑，我们在扫描初期，需要对目标网络的端口信息、系统信息、服务信息等进行收集，根据这些已知信息，合理进行后续的测试，而且在测试过程中也要按照由弱到强的顺序执行，如果首先进行高危测试，且目标系统存在相应漏洞，则会导致系统崩溃，测试无法继续进行。NASL 脚本在完成加载调度的过程中，需要经过三次处理，每次处理都定义了符合要求的链表结构对脚本及相关信息进行保存。

```
32行  include ("revisions-lib.inc")
```

添加引用的类库。

```
63行  script_id (10815) ;
```

添加 openvas ID。

但有些脚本会同时出现 script_cve_id（“CVE-2010-2963”、“CVE-2010-3067”），这代表添加 CVE ID。CVE（Common Vulnerabilities and Exposure）这类例子大家可以参看 deb_2126_1.nasl 脚本，如果该漏洞有 CVE 编号，可以在这个字段给出。同样在 freebsd_php5-gd.nasl 脚本里还出现了 script_bugtraq_id（33002），我们可以把 CVE 看作是一个词典，它提供了许多的交叉引用，现在大量公司都宣布他们的安全产品或数据库都和 CVE 相兼容，也就是能够利用 CVE 中漏洞名称同其他 CVE 兼容的产品进行交叉引用。通过 CVE 中为每个漏洞分配一个唯一的名称，在其他使用了这个名称的工具、网站、数据库以及服务中检索到相关信息，同时自身关于该漏洞的信息也能够被检索到。所以在网络安全评估时，利用 CVE 可以找出修补漏洞的措施，CVE 可以提供很好的指导。若要具体查询这些 ID 信息，可使用 grep 命令。

```
#grep -ir "script_bugtraq_id" /var/lib/openvas/plugins/
78行  script_family ("Web Servers") ;
```

设置脚本所属的族（family）。

NASL 对此没有明确规定，插件作者可以自己定义脚本所属的族。Openvas 使用的族包括：

- Debian Local Security Checks
- Fedora Local Security Checks
- Mandrake Local Security Checks
- HP-UX Local Security Checks
- Ubuntu Local Security Checks
- CentOS Local Security Checks
- Windows : Microsoft Bulletins
- FreeBSD Local Security Checks
- Red Hat Local Security Checks
- Solaris Local Security Checks
- CentOS Local Security Checks
- Web application abuses
- FreeBSD Local Security Checks
- Buffer overflow
- Firewalls
- Web Servers
- RPC
- General
- CISCO
- Databases

- Useless services
- Default Accounts
- Nmap NSE net
- Gain a shell remotely
- Malware
- Port scanners
- IT-Grundschutz
- SMTP problems
- Gentoo Local Security Checks
- SuSE Local Security Checks
- FTP
- Denial of Service
- Windows
- Service detection

实际上这些族就是主要的攻击种类。

下面继续看 79 行内容：

```
79行  script_copyright("Copyright (C) 2001 SecuriTeam, modified by Chris Sullo and Andrew Hintz");
```

设置脚本的版权信息

```
80行  script_dependendes("find_service.nasl", "httpver.nasl");
```

第 80 行脚本说明脚本依赖关系，如果要让 cross_site_scriping.nasl 正常运行，必须依赖 find_service.nasl"、"httpver.nasl"这两个脚本。

```
81行 script_require_ports("Services/www", 80);
```

第 81 行脚本表明执行此脚本所需的目标服务器的端口信息。

```
95行  port = get_http_port(default:80);获取服务器端口
123行  if(get_port_state(port))判断端口是否打开
134行  req = http_get(item:url, port:port);发送带有攻击性的请求
```

NASL 脚本主要是对攻击的描述，只是说明攻击的步骤，不是通常意义上的攻击。

```
135行   r= http_keepalive_send_recv(port:port, data:req, bodyonly: TRUE); 接收响应
165行   set_Kb_item(name:string("www/", port, "/generic_xss"), value:TRUE);
```

查看知识库中是否已存在此漏洞信息，知识库中保存了各类扫描所需相关信息，例如：

- 主机存活信息。
- 主机提供服务的信息。
- 端口扫描信息。
- 测试登录信息。

在脚本测试过程中，如果收集到有用的信息，将通过 set_Kb_item()函数，在知识库中增加相应条目；当需要对知识库中的相关信息进行调用时，则利用 get_Kb_item()函数进行读取。

2. 测试 NASL 脚本

将自己写的插件复制到 openvas 插件库目录/var/lib/openvas/plugins。

加载插件：

```
# openvassd
```

3. 重建插件库

```
# openvasmd --rebuild
```

注意：参数“rebuild”代表从一个正在运行的扫描器（openvassd）中重建数据库信息。

9.5.3 脚本加载过程

开发者需要理解的一个重点问题是，当启动 OpenVAS 时，由 openvassd 进程加载所有的脚本（/var/lib/openvas/plugins/下的*.nasl 脚本），存储为特定结构；根据客户端传递的配置信息，选取需要的脚本；最后根据选取脚本的 category 将脚本分组，组织脚本的执行顺序。

OpenVAS 关于脚本管理设计得非常合理，初始的加载脚本是保存的 argiist 链表内容，供后面进一步选取脚本及组织脚本执行顺序时调用，可有效减少内存的使用，提高工作效率。在脚本引擎的工作中，将脚本执行收集到的信息保存在知识库中，可有效避免重复扫描，减少不必要的资源浪费，提高工作效率。知识库中保存的信息包含各类扫描信息，例如：

- 主机存活信息（Host/dead）。
- 主机提供的服务信息（ftp/no_mkdir）。
- 端口扫描信息（Services/ftp）。
- 测试登录信息（SMB/login; SMB/password）。
- 用户检测时进行的系统设置信息。

在脚本执行过程中，如果收集到有用的信息，将通过 set_kb_item()函数，在知识库中增加相应的条目；如果需要对知识库中的相关信息进行调用，那么可以使用 get_kb_item()函数进行读取操作。

脚本在执行时如果有依赖的系统环境、端口状态等信息，会在知识库中查询，并根据查询结果做出合理的操作。例如：某脚本的执行需要目标主机的 80 端口打开，通过知识库查询，发现目标的 80 端口是关闭的，那么此脚本的执行将放弃，且该脚本对应的线程关闭；如果发

现目标的 80 端口处于打开状态，那么该脚本将顺利执行，对目标进行检测。通过对知识库中信息的查询判断，可有效提高脚本执行的效率，一些无意义的扫描将被过滤。

9.5.4　NASL 脚本介绍

通过对 OpenVAS 的分析，了解到其安全检测与评估的实现是以 NASL 脚本的调用与解析为基础的。NASL 脚本语言起初是针对网络安全扫描工具 Nessus 开发的，该脚本语言具有方便快捷，灵活性强，安全性高的特点，便于用户根据新出现的漏洞特性，写出有针对性的检测脚本。

NASL（Nessus Attack Scripting Language）脚本具有其特定的格式要求，其脚本内容可分为两个部分，第一个部分是脚本的注册信息，包含脚本名称、CVE-ID、漏洞描述信息、脚本类型、依赖的脚本或端口等信息。第二部分是针对相应漏洞的检测代码，此部分是脚本内容的核心，包含添加引用的库、脚本初始化、脚本执行及检测返回信息等操作。

NASL 脚本书写格式大致如下所示。

```
if(description)
{
/*注册信息：包含脚本名称、CVE-ID、漏洞描述信息、脚本类型，依赖的脚本或端口等*/
Script-name
CVE-ID;
exit(0)
}
/*脚本代码*/
```

NASL 脚本语言功能强大，语法类似于 C 语言，但是比 C 要简单，在使用 NASL 脚本语言时，不必考虑对象的类型、内存的分配与释放、变量声明等问题，只要认真设计漏洞的检测方法即可。由于漏洞扫描需在网络环境中进行，所以 NASL 语言提供了丰富的网络相关函数，包括套接字相关函数（用户可根据检测需要创建套接字，与目标主机进行通信）、原始报文处理函数（用户可根据扫描需求构建检测报文，并对返回信息进行分析，提取重要内容）。NASL 语言还提供了一些工具参数来简化检测脚本的书写，例如对目标主机系统、端口等信息的提取函数。

NASL 语言还提供了丰富的字符串处理函数。NASL 语言允许使用=、<和>等操作符以及正则表达式对字符串进行模式匹配，并且提供多种实现模式匹配功能的函数。NASL 语言包含字符串长度计算，内容复制，数据类型转换、大小写转换等功能函数，便于脚本编写。

9.6　OpenVAS 脚本分析

OpenVAS 系统利用 NASL 脚本对目标主机进行安全检测与评估，每个脚本都有其特定的

功能，脚本的代码内容确定了其功能的实现过程，按照不同的业务需要选取脚本，即可实现符合特殊要求的安全扫描。

9.6.1 OpenVAS 脚本类别

截至 2015 年 9 月，OpenVAS 系统所包含的脚本已经超过 4 万多个，所有漏洞库脚本名称记录在 alienvault. vuln_nessus_plugins_feed 表中。根据测试的漏洞类型、漏洞存在位置等信息，将脚本分类，现有脚本族共有 50 种。详细插件信息读者可以到网址 http://www.freebsdports.info/ports/security/openvas-plugins.html 查看。根据脚本族的分类可知，OpenVAS 可以对网络平台、应用平台以及 Web 应用进行检测。

应用平台包括 Web 应用所使用的操作系统、服务器、数据库等。Web 应用只包含为前面所涉及的网络平台与应用平台提供服务的 Web 应用，并不包含第三方开发的 Web 应用。详细的 OpenVAS 功能范围如表 9-4 所示，其中红色部分为 OpenVAS 不能进行检测的第三方 Web 应用。

表 9-4　OpenVAS 功能范围

应用平台	Web 应用	Apache、Nginx、Tomcat、MS IIS 等
	数据库	数据库包括 IBM DB2 、MySQL、Oracle 等
	中间件	Weblogic、Lotus
	操作系统	各种 Unix/Linux/Windows 、Vmware ESX 系统
	浏览器	Safari
网络平台	防火墙	Cisco、CheckPoint、Clamav 等
	网络设备	Cisco、3COM、Nortel、Bluecoat 设备
	通信协议	P2P、RPC、SNMP 等

表 9-4 对 OpenVAS 进行安全检测的对象进行分类 并由此分析 OpenVAS 的功能 OpenVAS 安全评估系统可对目标主机的网络平台、应用平台及 Web 应用进行安全扫描，但扫描前，需对扫描的主机相关信息进行收集，信息收集的工作则由 Port Scanners 脚本族中一些脚本完成。

Port Scanners 脚本族共含有 15 个脚本，包含多种不同类型的扫描方式，OpenVAS 默认扫描配置中，选取其中 2 个脚本 ping_host.nasl 和 nmap.nasl 实现端口扫描工作（脚本位于 /var/lib/openvas/plugins 中）。其中 ping_host.nasl 脚本用来判断目标主机是否存活，nmap.nasl 脚本则调用 nmap 来对目标主机进行详细的扫描，收集相关信息。

9.6.2 同步 OpenVAS 插件

如果启用 OSSIM 的漏洞扫描功能，还需要手工升级 OpenVas 插件库，同步方式分为在线和离线两种，如图 9-69 所示为采用离线升级方式。

```
alienvault:/usr/share/ossim/scripts/vulnmeter# openvas-nvt-sync --wget
[i] Using GNU wget: /usr/bin/wget
[i] Configured NVT http feed: http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
[i] Downloading to: /tmp/openvas-nvt-sync.lc7sR8McH5/openvas-feed-2014-04-15-27764.tar.bz2
--2014-04-15 12:55:57--  http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
Resolving www.openvas.org... 5.9.98.186
Connecting to www.openvas.org|5.9.98.186|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14741386 (14M) [application/x-bzip2]
Saving to: `/tmp/openvas-nvt-sync.lc7sR8McH5/openvas-feed-2014-04-15-27764.tar.bz2'

49% [==========================>                           ] 7,240,629    130K/s  eta 92s
```

图 9-69　同步插件

更新插件之后，需重启 openvas-scanner 服务，由于插件数量众多，同步过程比较长。

1. 在线方式同步

在 OSSIM 控制台下，输入以下命令，效果如图 9-70 所示。

```
#openvas-nvt-sync
```

此脚本通过使用 rsync 及 md5sum 命令在 rsync://rsync.openvas.org 升级更新程序和证书。这一步完成之后，将下载 NVT Feed 到/var/lib/openvas/plugins/中。NVT（Network Vulnerability Test）代表网络漏洞测试脚本。

```
report_formats/bef0180e-6960-11e1-931f-001f29e71d12.asc
         198 100%    0.20kB/s    0:00:00 (xfer#27421, to-check=5/69163)
report_formats/bef0180e-6960-11e1-931f-001f29e71d12.asc.asc
         198 100%    0.20kB/s    0:00:00 (xfer#27422, to-check=4/69163)
report_formats/d5da9f67-8551-4e51-807b-b6a873d70e34.asc
         197 100%    0.20kB/s    0:00:00 (xfer#27423, to-check=3/69163)
report_formats/d5da9f67-8551-4e51-807b-b6a873d70e34.asc.asc
         197 100%    0.20kB/s    0:00:00 (xfer#27424, to-check=2/69163)
report_formats/f5c2a364-47d2-4700-b21d-0a7693daddab.asc
         197 100%    0.20kB/s    0:00:00 (xfer#27425, to-check=1/69163)
report_formats/f5c2a364-47d2-4700-b21d-0a7693daddab.asc.asc
         197 100%    0.20kB/s    0:00:00 (xfer#27426, to-check=0/69163)

sent 1006625 bytes  received 60952467 bytes  17372.52 bytes/sec
total size is 184906490  speedup is 2.98
[i] Checking dir: ok
[i] Checking MD5 checksum: ok
You have new mail in /var/mail/root
alienvault:~#
alienvault:~#
alienvault:~#
alienvault:~# cd /var/lib/openvas/plugins/
alienvault:/var/lib/openvas/plugins# ls *.nasl|wc
  28832   28832  853844
alienvault:/var/lib/openvas/plugins# ls *.nasl.asc|wc
  28832   28832  969172
alienvault:/var/lib/openvas/plugins# ls *.nasl.asc|wc_
```

图 9-70　在线同步

更新时长由更新库的总量和带宽决定，所有升级漏洞检测脚本文件存放在/var/cache/openvas/中。切勿强行中断升级过程。

- *.nasl.desc，10524 个。

- *.nasl.nvti，28832 个。

2. 更新插件

下载完所有升级包后，建议停止所有 openvas 进程。

```
#/etc/init.d/openvas-administrator stop
#/etc/init.d/openvas-manager stop
#/etc/init.d/openvas-scanner stop
```

在 Web UI 下不能有任何漏洞扫描任务，为了确保成功，建议读者输入如下命令查验：

```
#ps ax |grep openvas                \\*查看系统是否还有 openvas 进程存在。
```

下一步，进行插件升级操作。

```
#/usr/share/ossim/scripts/vulnmeter/updateplugins.pl update wget
```

此行命令相当于“/usr/bin/openvas-nvt-sync --wget”执行的效果，正常执行完该命令之后，提示“Framewor profile has been found”、“Sensor profile has been found”、“BEGIN - PERFORM UPDATE”。接下来在开始更新数据库之前，需要重建 OpenVAS NVT 缓存，该（Rebuilding NVT cache）过程后台工作时间较长（切勿强行中断），后台由 openvassd 进程负责执行，大家一定要耐心等待进程执行完成，切勿强行中断。

最后启动 Openvas 服务。

```
#killall openvassd
#sleep 15
#/etc/init.d/openvas-scanner start
#/etc/init.d/openvas-manager start
#/etc/init.d/openvas-administrator restart
```

以下问题需要注意：如果执行升级命令后，出现“No Sensors Found at …”，表示 Sensor 没有和 Server 建立连接，这时需要查找双方的配置问题。下面我们查看实际操作的效果，如图 9-71 所示。

```
alienvault:~# /usr/share/ossim/scripts/vulnmeter/updateplugins.pl update wget
2014-08-08 15:32:40   Framework profile has been found...
2014-08-08 15:32:40   Skipping ip 10.32.x.y
No sensors found at /usr/share/ossim/scripts/vulnmeter/updateplugins.pl line
313.
You have new mail in /var/mail/root
```

```
alienvault:~# /usr/share/ossim/scripts/vulnmeter/updateplugins.pl  update wget

2014-04-15 13:36:01   Framework profile has been found...
2014-04-15 13:36:02   Deleting all tasks in 192.168.120.130 ...
2014-04-15 13:36:03   updateplugins: executing update-plugins
2014-04-15 13:36:03   BEGIN  - PERFORM UPDATE
2014-04-15 13:36:03   /usr/sbin/openvas-nvt-sync --wget >> /tmp/update_scanner_plugins_rsync.log
2014-04-15 13:37:26   Fixing OpenVAS Plugins...
2014-04-15 13:39:12   Rebuilding NVT cache...
2014-04-15 14:13:51   FINISH - PERFORM UPDATE [ Process took 2268 seconds ]
2014-04-15 14:13:51   updateplugins: configured to not migrate DB
2014-04-15 14:13:51   BEGIN  - DUMP PLUGINS
2014-04-15 14:13:51   /usr/bin/omp -h 192.168.120.130 -p 9390 -u 'ossim' -w 'ossim' -iX - < /usr/sha
re/ossim/www/tmp/omp_command_6120896105.xml > /usr/share/ossim/www/vulnmeter/tmp/plugins.xml 2>&1
2014-04-15 14:21:01   FINISH - DUMP PLUGINS [ Process took 430 seconds ]
2014-04-15 14:21:01   BEGIN  - IMPORT PLUGINS..FAILED - CREATE TABLE vuln_plugins (
                     id int NOT NULL,
                     oid varchar(50) NOT NULL,
                     name varchar(255),
                     family varchar(255),
                     category varchar(255),
                     copyright varchar(255),
                     summary varchar(255),
                     description blob,
                     version varchar(255),
                     cve_id varchar(255),
                     bugtraq_id varchar(255),
                     xref blob,
                     cvss_base varchar(50) NOT NULL,
                     primary key (id))
Lost connection to MySQL server during query

DBD::mysql::st execute failed: MySQL server has gone away at /usr/share/ossim/scripts/vulnmeter/upda
teplugins.pl line 642.
You have new mail in /var/mail/root
alienvault:~#
```

图 9-71　更新插件

插件扩展名为“nasl.asc”，系统中指定下载位置在/var/lib/openvas/plugings/目录中，在目前最新版的 OSSIM 系统中提供了图形化升级命令，如图 9-72 所示。

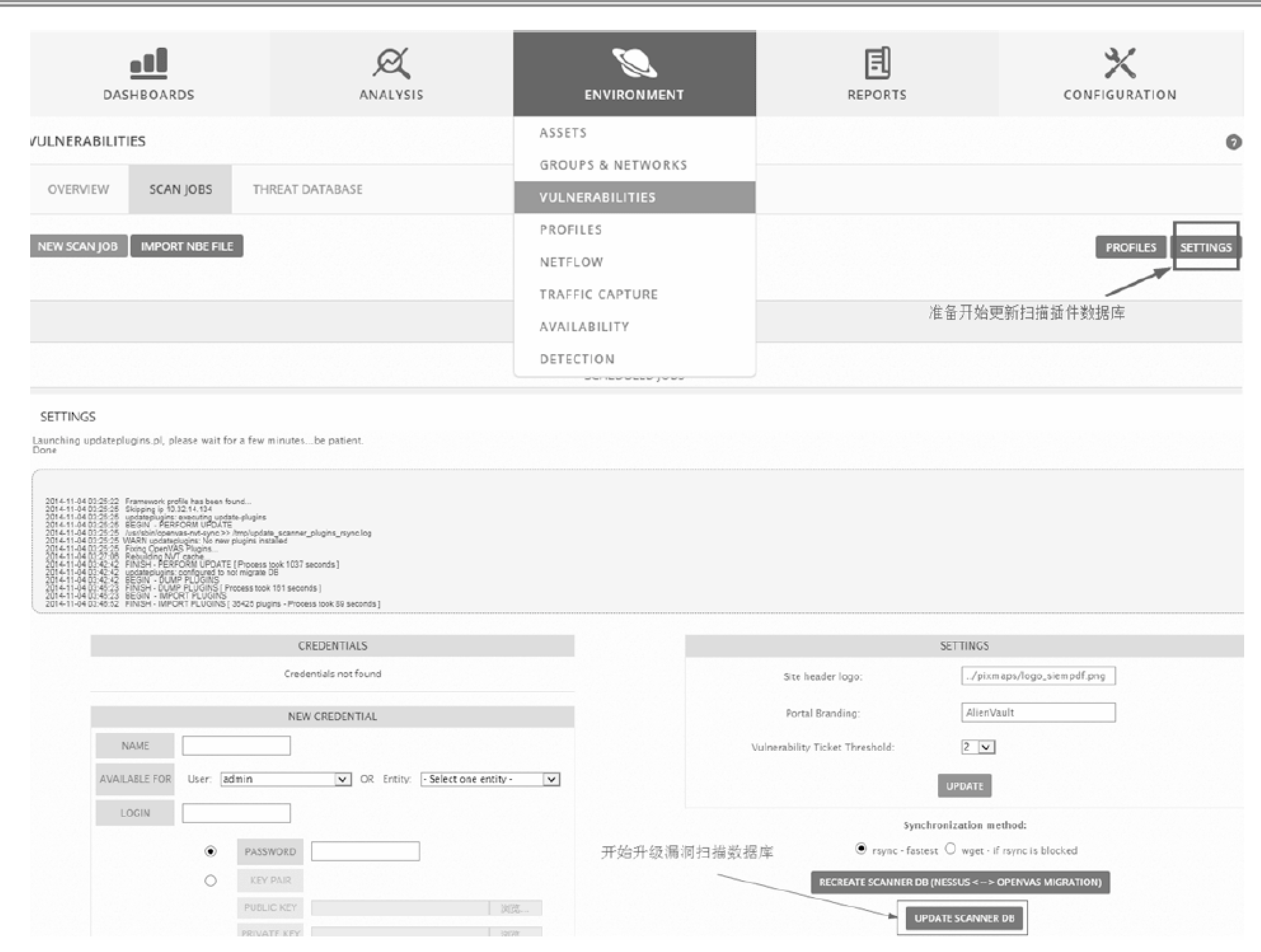


图 9-72　升级设置

当单击 UPDATE SCANNER DB 按钮后，系统在后台执行“/usr/bin/rsync -ltvrP --delete

--exclude private/ rsync://feed.openvas.org:/nvt-feed /var/lib/openvas/plugins”指令并开始下载、更新漏洞数据库，升级完后可查看“THREAT DATABASE”信息，其存储在数据库 alienvault 的表 vuln_nessus_plugins 中，截至 2015 年 4 月，其总数为 38296。

3. USM 5.0 系统中的升级

OSSIM USM5.0 中升级过程得到了简化，只要在更新后，单击 SETTINGS 设置按钮，打开设置窗口并单击修复扫描器数据库按钮即可，如图 9-73 所示。升级过程中勿关闭此窗口。

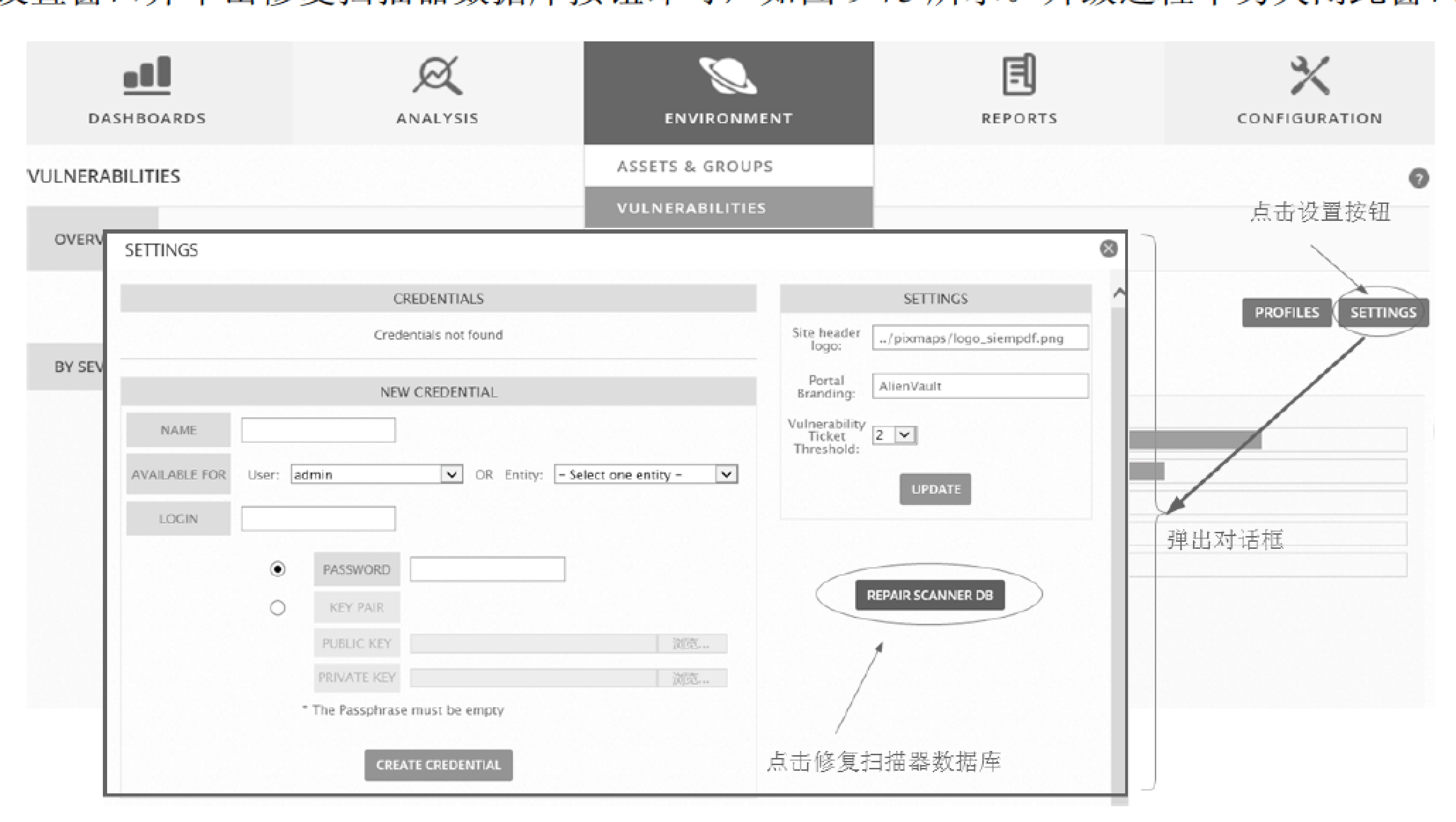


图 9-73　USM5.0 下升级方法

4. 离线方式同步

```
#openvas-nvt-sync --wget
```

执行该命令，系统会将 openvas-feed-日期.tar.bz2 包临时下载到/tmp 目录下自行解压后，自动删除该压缩包，当所有新插件完成下载后，可以利用以下命令更新插件，如图 9-74 所示。

```
#wget http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
```

```
--2015-02-24 22:19:56--  (try: 5)  http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
Connecting to www.openvas.org|5.9.98.186|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17982584 (17M) [application/x-bzip2]
Saving to: `openvas-nvt-feed-current.tar.bz2'

66% [========================================>                    ] 12,030,000   345K/s  eta 16s
```

图 9-74　下载离线升级包

```
#tar -jxf openvas-nvt-feed-current.tar.bz2 -C /var/lib/openvas/plugins/nvt/
```

然后，解压至/var/lib/openvas/plugins/nvt/目录（注意：如果没有 nvt 目录，则需要手工创建），下面命令开始更新插件：

```
#perl /usr/share/ossim/scripts/vulnmeter/updateplugins.pl migrate
#/usr/share/ossim/scripts/vulnmeter/updateplugins.pl update wget
```

要使其生效，最后就要重启 openvas 服务，输入以下 3 条命令：

```
#/etc/init.d/openvas-manager restart
#/etc/init.d/openvas-administrator restart
#/etc/init.d/openvas-scanner restart
```

5. Openvas 的同步问题

在 OpenVAS 扫描时，涉及两种同步分别是 rsync 和 wget，使用 wget 进行两个服务器间单向文件同步，之前曾经使用过 FTP Sync 进行网站同步，结果发现 wget 已有类似功能，而且更加简单，例如：

```
#wget -m --no-remove-listing -nH -P /home/xxx/bak/
ftp://username:password@www.xxx.com/*
```

最后 ftp 路径后面有个星号“*”，否则 wget 只会下载生成一个目录列表文件 index.html，而不会同步文件。

参数含义：

- -m：打开单向镜像。
- --no-remove-listing：不生成目录列表文件 index.html。
- -nH：不创建以主机名命名的目录。
- -P：下载保存路径。

9.7 漏洞扫描实践

9.7.1 漏洞库

目前许多欧美的国际安全组织都按照自己分类准则建立了各自的数据库，其中主流是 CVE 和 XForce，它的好处是当网络出现安全事故，入侵检测系统（IDS）产生警报时，像 CVE 这类标准的系统脆弱性数据库的建设工作就显得极为重要。目前在中国国家计算机网络应急处理协调中心（CNCERT/CC）领导下，国内迅速组建了自己的 CVE 组织——CNCVE，CNCVE 的组建目的就是建设一个具有中国特色的，能为国内广大用户服务的 CVE 组织。但是并不是说拥有了 CVE 就包括了所有漏洞问题，除了这些开放的脆弱性数据库外，还应该存在大量的没有对公众开放的脆弱性数据库。

（1）CVE

CVE（Common Vulnerabilities and Exposures）由美国国安局赞助。CVE 所指定的标准已成为国际标准，以方便独立的脆弱性数据库和不同安全工具彼此之间能够更好地共享数据。CVE 的标准命名方式是由“CVE”、时间和编号三部分组成。例如，命名为“CVE-2004-1161”的条目表示 2004 年第 1161 号脆弱性，如图 9-75 所示。

VULNERABILITIES

OVERVIEW　SCAN JOBS　THREAT DATABASE

SEARCH RESULTS FOR THIS CRITERIA

START DATE	END DATE	KEYWORDS	CVE ID	FAMILY	RISK FACTOR
All	All	All	CVE-2014	All	High

ID	RISK	DEFINED ON	THREAT FAMILY & SUMMARY	CVE ID
103884		2014-05-21 17:23:54	VMware Local Security Checks - Checks for installed patches.	CVE-2014-1207 CVE-2014-1208 CVE-2014-1211
103888		2014-05-21 17:23:54	CISCO - Check the NX OS version	CVE-2014-0677
103889		2014-05-21 17:23:54	CISCO - Check the NX OS version	CVE-2014-0676
103892		2014-05-21 17:23:54	Web application abuses - Check the version	CVE-2014-1671
103904		2014-05-21 17:23:54	Web application abuses - Determine if it is possible to read a local file	CVE-2014-1922 CVE-2014-1923 CVE-2014-1924 CVE-2014-1925
103913		2014-05-21 17:23:54	Web application abuses - Determine if it is possible to inject sql code	CVE-2014-1945
103920		2014-05-21 17:23:54	Web application abuses - Determine if it is possible to execute a command	CVE-2014-1903

图 9-75　列出漏洞库信息

OpenVAS 中含有扫描器，在漏洞扫描时，首先由扫描器对目标主机进行扫描，获得主机系统信息后，和系统的漏洞库进行匹配，查找出可能存在的漏洞，扫描器所能获得的信息主要包括操作系统名称、版本及系统提供的服务。

为了统一命名，漏洞库中必须为每条漏洞信息提供统一编号，该图中显示的 CVE-2014-0160，就是 CVE 的编号。我们知道 OpenSSL 是 Apache 和 Nginx 网络服务器的默认安全协议，此外大量的操作系统、电子邮件和即时通信系统也采用 OpenSSL 加密用户数据通信。在 2014 年 4 月初，研究者发现 OpenSSL v1.0.1 到 1.0.1f 的密码算法库中发现了一个非常严重 BUG（CVE-2014-0160），该 BUG 允许攻击者读取存在 BUG 的系统的 64KB 处理内存，暴露加密流量的密钥、用户的名字和密码，以及访问的内容。

CVE 的内容是 CVE 编辑委员会的合作努力的成果。这个委员会的成员来自于许多安全相关的组织，如软件开发商、大学研究机构、政府组织和一些优秀的安全专家等，而且 CVE 可以免费阅读和下载。

（2）OSVDB

OSVDB（Open Source Vulnerability Database）是由一个社团组织创立并维护的独立开源的数据库。它最早是在 2002 年的 Black Hat 和 Defcon 安全会议上提出的一项服务，它提供了一个独立于开发商的脆弱性数据库实现方案。和 CVE 一样，OSVDB 数据库也是开源并且免费，它由安全爱好者来维护，向个人和商业团体免费开放。两者的差异在于 CVE 提供标准名

称，可以通俗理解为数据字典，而 OSVDB 为每一条脆弱性提供了详尽的信息，OSVDB 需要参考 CVE 的名称。在 OSVDB 漏洞模型中有以下概念比较重要，读者需要了解：

- 漏洞 ID：OSVDB 分配了唯一的漏洞 ID 编号。
- 漏洞标题：它是对问题性质和受影响产品的一般性概括。
- 公布日期：漏洞首次发布的日期，OSVDB 还细分为发现日期（漏洞被公共邮件列表公布的日期）、利用日期（相关的漏洞利用手段首次被发现的日期），以及解决日期（厂家第一次公开发布漏洞补丁的日期）。
- 分类：OSVDB 提供自己对漏洞的分类，包括攻击类型、破坏性、本地漏洞还是远程漏洞和那些服务相关联等。
- 解决方案：对漏洞的修复说明。
- 受影响产品：这部分包含可能受漏洞影响的厂家列表。
- 引用：表示本网站收集的 URL 信息和第三方提供的漏洞信息。在 OSVDB 中，每条引用需要注释引用类型。OSVDB 提供了指向其他主流漏洞数据库的引用，例如 Security Focus、Secunia、CVE、US-CERT 以及 VUPEN；对漏洞利用的引用如 Metasploit、Milw0rm（知名的黑客攻防小组）；扫描工具签名引用，如 Nessus Script ID、Snort Signature ID、OVAL ID、Nikto Item ID 等。

（3）BugTraq

BugTraq 是由 Security Focus 管理的 Internet 邮件列表，现在已被赛门铁克公司收购。在安全界，BugTraq 相当于一个权威的专业杂志。大多数安全技术人员订阅 Bugtraq，因为这里可以抢先获得关于软件、系统漏洞和缺陷信息。

9.7.2 常见漏洞发布网站

通常漏洞研究都和英文资料打交道，所以阅读下面的内容需要具有一定英文水平，这些站点为安全分析人员提供了参考依据：

- 国家信息安全漏洞共享平台：http://www.cnvd.org.cn/
- 乌云：http://www.wooyun.org/bugs/
- 国际权威漏洞库：http://www.securityfocus.com
- IBM 网络安全漏洞库：http://xforce.iss.net
- 俄罗斯知名安全实验室：http://en.securitylab.ru
- 安全厂商绿盟科技：http://www.nsfocus.net
- 安全焦点：http://www.securityfocus.com/archive/ /
- 国际权威漏洞库：http://osvdb.org
- http://cve.mitre.org/
- http://www.exploit-db.com
- http://www.milw0rm.cn/ldong/

9.7.3 手动更新 CVE 库

如何更新 OSSIM 系统中的 CVE 数据库呢？在本书第 1 章中介绍了 OSSIM 计划任务，其中每天执行的计划任务包含 alienvault-passvulnupdate，在这个脚本中包含升级方法，但读者还可通过手动方式升级，操作命令如下：

```
#openvas-scapdata-sync
```

该命令从 nvd.nist.gov 站点下载最新的 XML 文件，升级文件位于/var/lib/openvas/scap-data/目录，其格式为 nvdcve-2.0-<年>.xml，以年为单位。

openvas-scapdata-sync 命令首先下载 xml，然后进行自动更新，某些情况下，下载了全部文件并更新后，若发现没有更新，则需要执行以下脚本：

```
#/usr/share/ossim/scripts/vulnmeter/updateplugins.pl
```

9.7.4 采用 OpenVAS 扫描

OSSIM 系统通过 OpenVAS 进行漏洞扫描，它提供了大量免费扫描插件（NVT ，Network Vulnerability Test），OSSIM 系统中包括四个 OpenVAS 系统服务，各自承担不同的用途。如果选用较新的 OSSIM 4.15 系统，最低内存配置为 16GB。从部署扫描器的模式上看，可分为单独部署和分布式部署两种方式。

单独部署即单一节点扫描，与传统 Nessus 或 X-scan 扫描器相似，分布式部署则通过在每个网络单元的 Sensor 上，部署独立的扫描引擎，对该网络单元的 IT 资产进行扫描，在管理中心部署安全漏洞管理控制平台，对所有的扫描引擎进行统一管理，同时所有引擎的扫描结果，全部返回控制平台进行汇总，最终实现全网漏洞的统一呈现、分析和管理，所以它和资产管理又紧密相连。下面重点讲述 OpenVAS 的配置细节。

1. 主要进程和配置

Openvas 配置文件在/etc/openvas/目录中，OSSIM 系统中已配置好 Openvas，默认创建管理员用户 admin，所以不必像上面 Nessus 那样有比较复杂的配置过程。服务器层包含以下三个组件：

- Openvas Manager，管理程序，负责与客户端程序通信，完成扫描任务、配置信息、用户授权及检测报告等工作并存储这些信息，它是整个漏洞扫描平台的中央控制服务，它的 OMP（Openvas Management Protocol）基于 XML，用于设置扫描计划和生成测试报告，保存扫描结果等，Openvas Manager 程序默认为 TCP 9390 端口。
- Openvas Scanner，Openvas 的扫描器，实际执行各种网络漏洞扫描测试（NVT）的主服务，包括 NVT 的订阅和更新，还负责调用各种漏洞检测插件，默认为 TCP 9391 端口。
- Openvas Administrator，负责与 openvas-manager 通信，完成用户和配置管理等操作，默认监听地址为 127.0.0.1，端口为 TCP 9393。

openvas-manager、openvas-scanner 会在 OSSIM 系统启动时自动启用，其余两个服务根据需要手动启动。下面我们在 OSSIM 系统中查看这两个启动文件内容。

```
#cd /etc/default
#cat openvas-manager
DATABASE_FILE=/var/lib/openvas/mgr/task.db
MANAGER_ADDRESS=0.0.0.0
MANAGER_PORT=9390
SCANNER_ADDRESS=127.0.0.1
SCANNER_PORT=9391
#cat openvas-scanner
SCANNER_ADDRESS=0.0.0.0
SCANNER_PORT=9391
```

通过这个配置文件发现 openvas-manager 监听端口为 9391。

```
#cat openvas-administrator
ADMINISTRATOR_ADDRESS=127.0.0.1
ADMINISTRATOR_PORT=9393
USER_DATA=/var/lib/openvas/users
SCANNER_CONFIG=/etc/openvas/openvassd.conf
SYNC_SCRIPT=/usr/sbin/openvas-nvt-sync
```

通过这个配置文件，我们发现 openvas-administrator 监听端口为 9393，了解这些关键进程的监听端口，对今后我们进行故障排除非常有帮助。

2. OSSIM 下操作 Openvas 方法

必要条件：首先必须具有足够大的内存（16GB 以上），其次升级补丁工作建议在系统刚架设完毕就做，然后升级 Openvas 插件。最后确保 Sensor 传感器为可用状态，Nmap 进程运行正常。

扫描前要确保服务器和客户机的时钟要保持一致。

操作步骤概括如下：

- 定义扫描目标，可以是 IP 地址、网段地址;
- 创建新任务，选择扫描目标、扫描方式;
- 运行新创建的扫描任务;
- 等待扫描结束，然后查看扫描报告。

以 OSSIM 4.3 平台为例，操作过程说明如下：

（1）选择 Anaylysis→Vulnerabilities ，然后单击 New scan job 按钮，开始创建一个新的扫描任务。

（2）从上到下依次输入任务名称，例如“Server1”，选择关联引擎，如果是分布式系统就要选择相应的传感器，然后选择扫描方式选项，包括立即执行、每天/每周执行，系统提供了非常详细的计划任务列表，如图 9-76 所示。

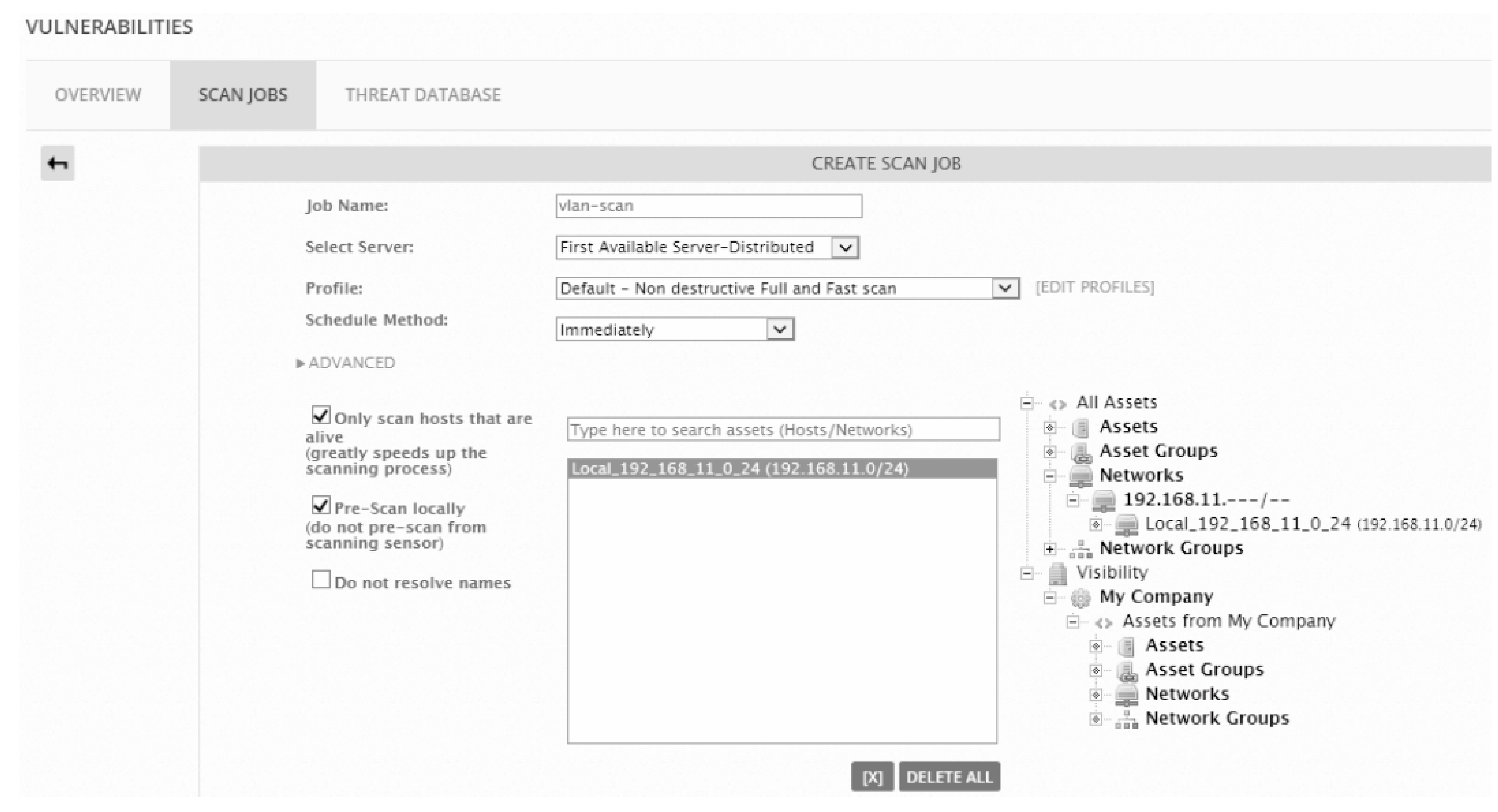


图 9-76　扫描网段

建议不要扫描过多的主机（以本书第 1 章推荐的服务器配置为基准，建议每次扫描主机数量，以不超过 30 台为宜，在系统/etc/openvas/openvassd.conf 配置文件中，定义最大主机数为 5 台，每台主机同时检查插件数为 10 条），如图 9-77 所示。

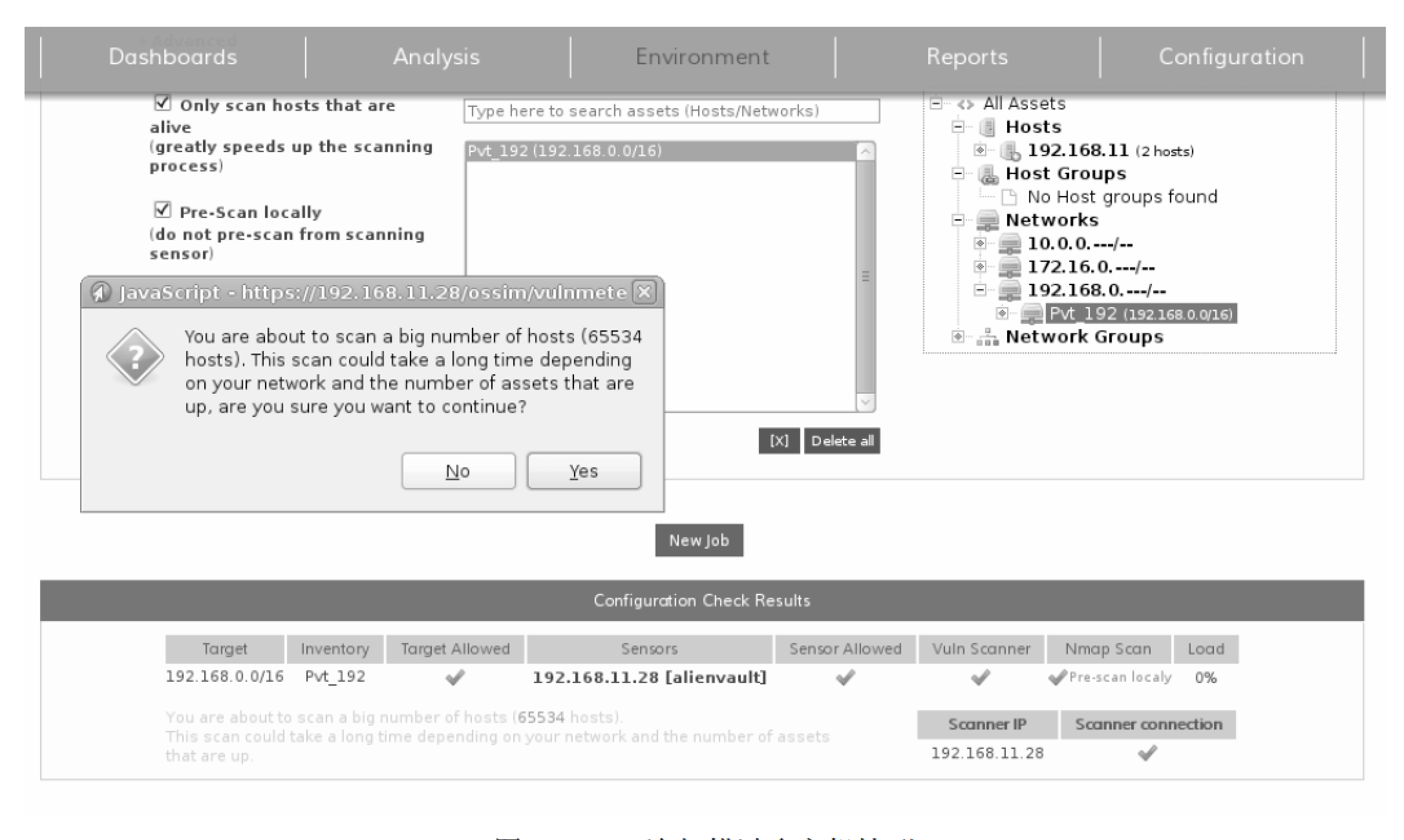
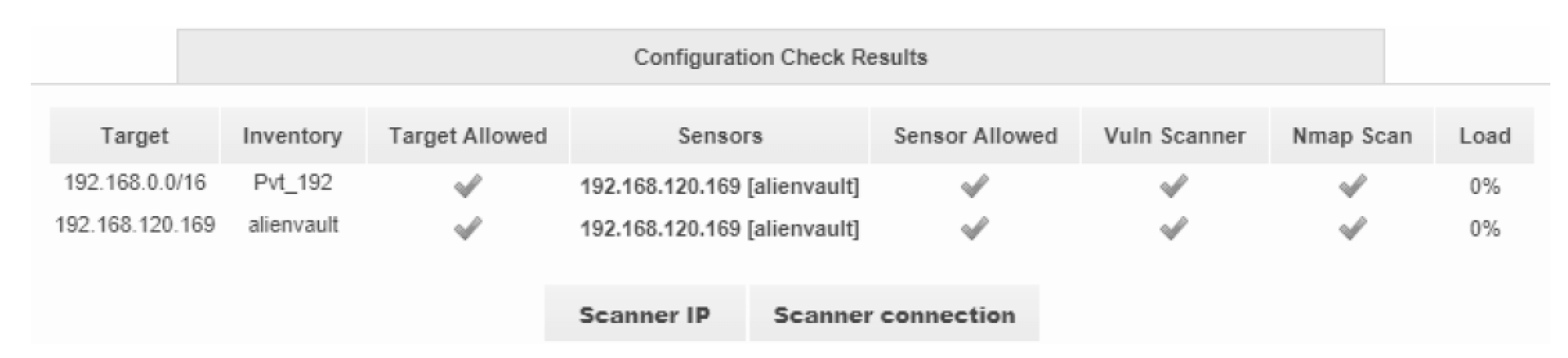


图 9-77　一次扫描过多主机情形

（3）最后对任务进行扫描检测以确保以下要求：扫描检查结果包含目标 IP 和主机名，传感器是否启用，漏洞扫描库是否准备好，Nmap 是否可用这几个参数，如图 9-78 所示。



Configuration Check Results

Target	Inventory	Target Allowed	Sensors	Sensor Allowed	Vuln Scanner	Nmap Scan	Load
192.168.0.0/16	Pvt_192	✔	192.168.120.169 [alienvault]	✔	✔	✔	0%
192.168.120.169	alienvault	✔	192.168.120.169 [alienvault]	✔	✔	✔	0%

Scanner IP　Scanner connection

图 9-78　扫描准备工作

选取系统资源时，注意要从 Assets 资源池中选取，而不能手工输入 IP 地址的方式。OSSIM 4.6 以后的系统 Web UI 发生较大变化，其 Web 漏洞扫描菜单路径为：Environment→Vulnerabilities，如图 9-79 所示。

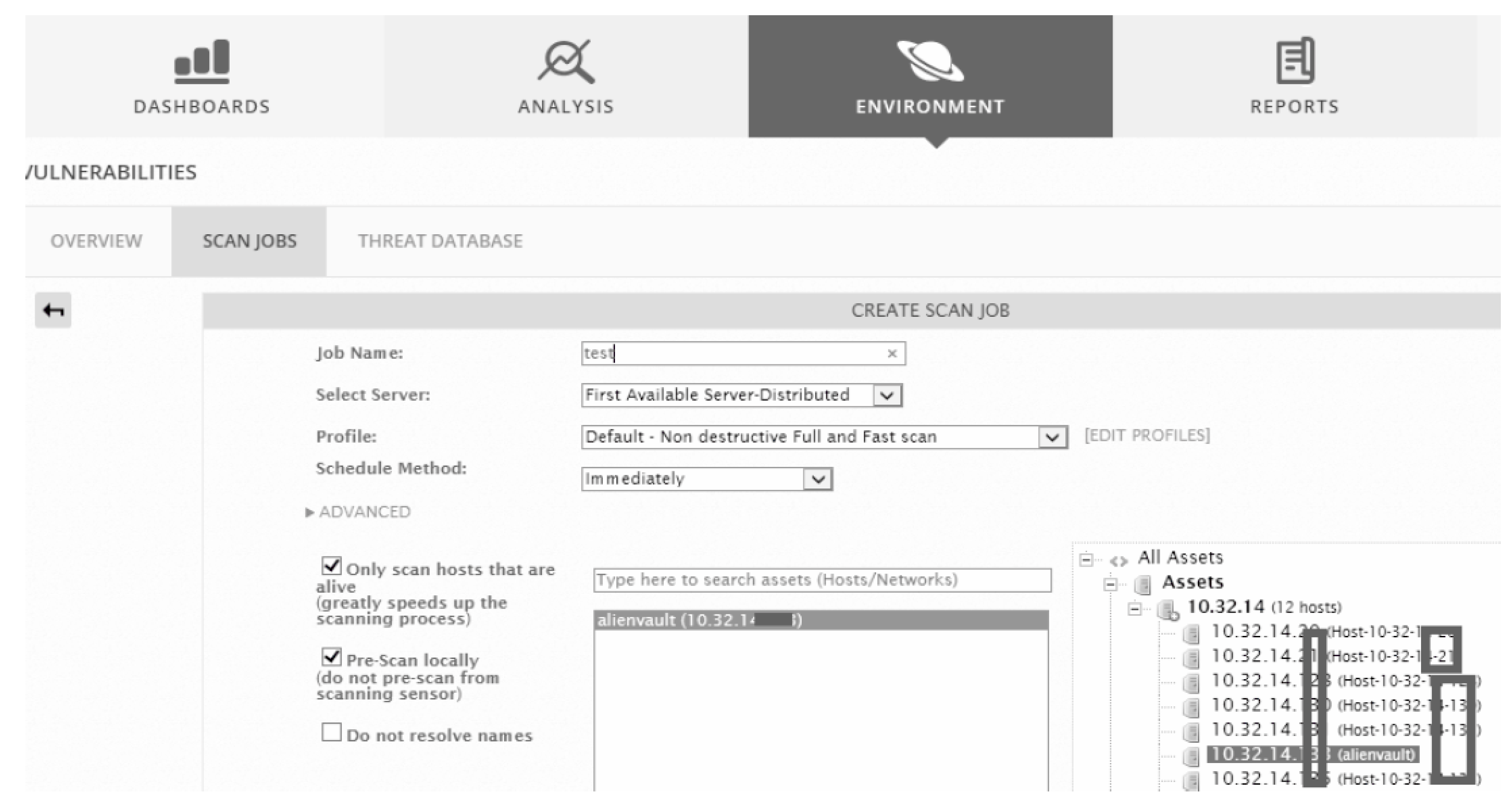


图 9-79　OSSIM 4.6 漏洞扫描设置

OpenVAS 提供了三种策略：

- Deep-Non destructive Full and Slow scan（非破坏性的全面和慢扫描）。
- Default-Non destructive Full and Fast scan（非破坏性全面和快扫描）。
- Ultimate-Full and Fast scan including Destructive tests（极限方式、全面和快扫描，包含破坏性测试速度最慢，慎用）。

如图 9-80 所示为选择 Sensor 进行扫描。

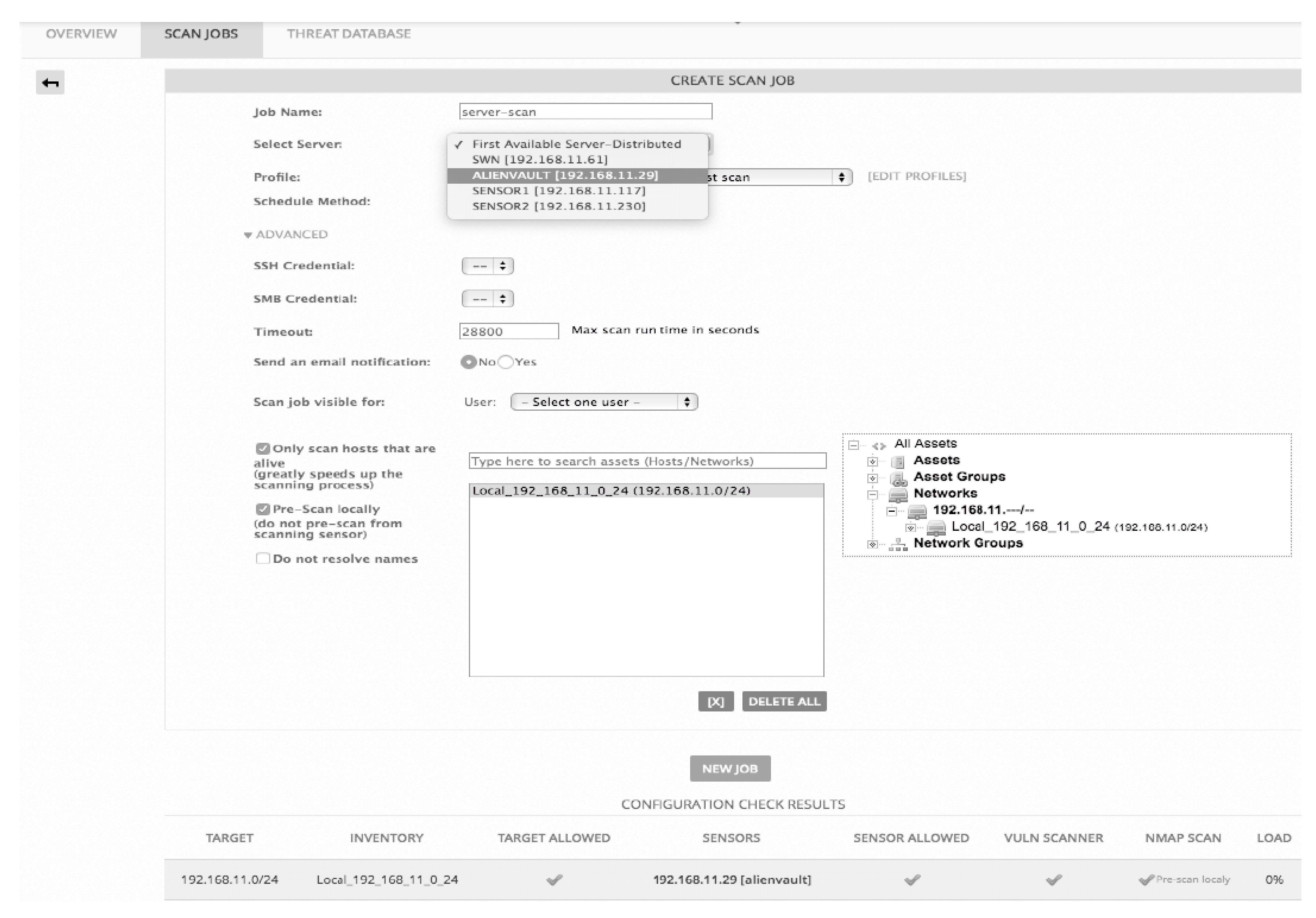


图 9-80　选择 Sensor 进行扫描

无论哪个版本的 OSSIM 扫描时，都会消耗服务器的大量资源。通常情况下，建议扫描机器不宜过多，每个任务以 25 台左右为宜，可根据机器配置情况做适量增减。在配置文件/etc/openvas/openvassd.conf 中可以适当改变该值。

9.7.5　扫描过程

在 OSSIM 中进行漏洞扫描，首先保证内存足够。OpenVas 管理进程和扫描进程均正常启动，端口处于监听状态。正常情况下扫描从开始经过四个状态，分别是 Schedued、Requested、Running、Commpleted，如该变迁过程出现“Failed”或“Timeout”都表示扫描过程中出现异常，如图 9-81 所示。

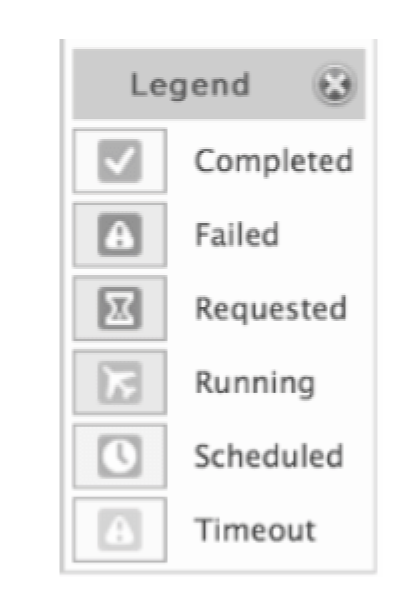


图 9-81　漏洞扫描状态标识

扫描结果分类：

（1）按严重程度分类

Openvas 用 5 种颜色对漏洞进行分级显示，并将不同种类扫描漏洞数量用饼图表示出来，各种颜色代表含义如下：

- Serious：用紫色表示，表示非常严重的漏洞，将会影响关键业务的紧急威胁需要立即修复。

- Hight：用红色表示，表示高度危险，影响关键业务，非直接威胁，对于以上两种高危漏洞需要再进行渗透测试工作。
- Medium：用橘红色表示，代表中度危险，影响业务的非直接威胁。
- Low：用黄色表示，代表低危险性。
- Info：用淡黄色表示，代表普通信息。

通常我们用 Nessus 扫描系统后得到了各种类型、数量众多的扫描结果，大家开始发愁，这么多的信息，该怎样把它们整理并分类，更加直观地展示出来？在图 9-82 中，OSSIM 已为你考虑到这一点，它使用 MySQL 数据库来保存渗透测试过程中获取的各种数据，并通过饼图和柱状图把这些存储在数据库中的数据以可视化方式展示出来。

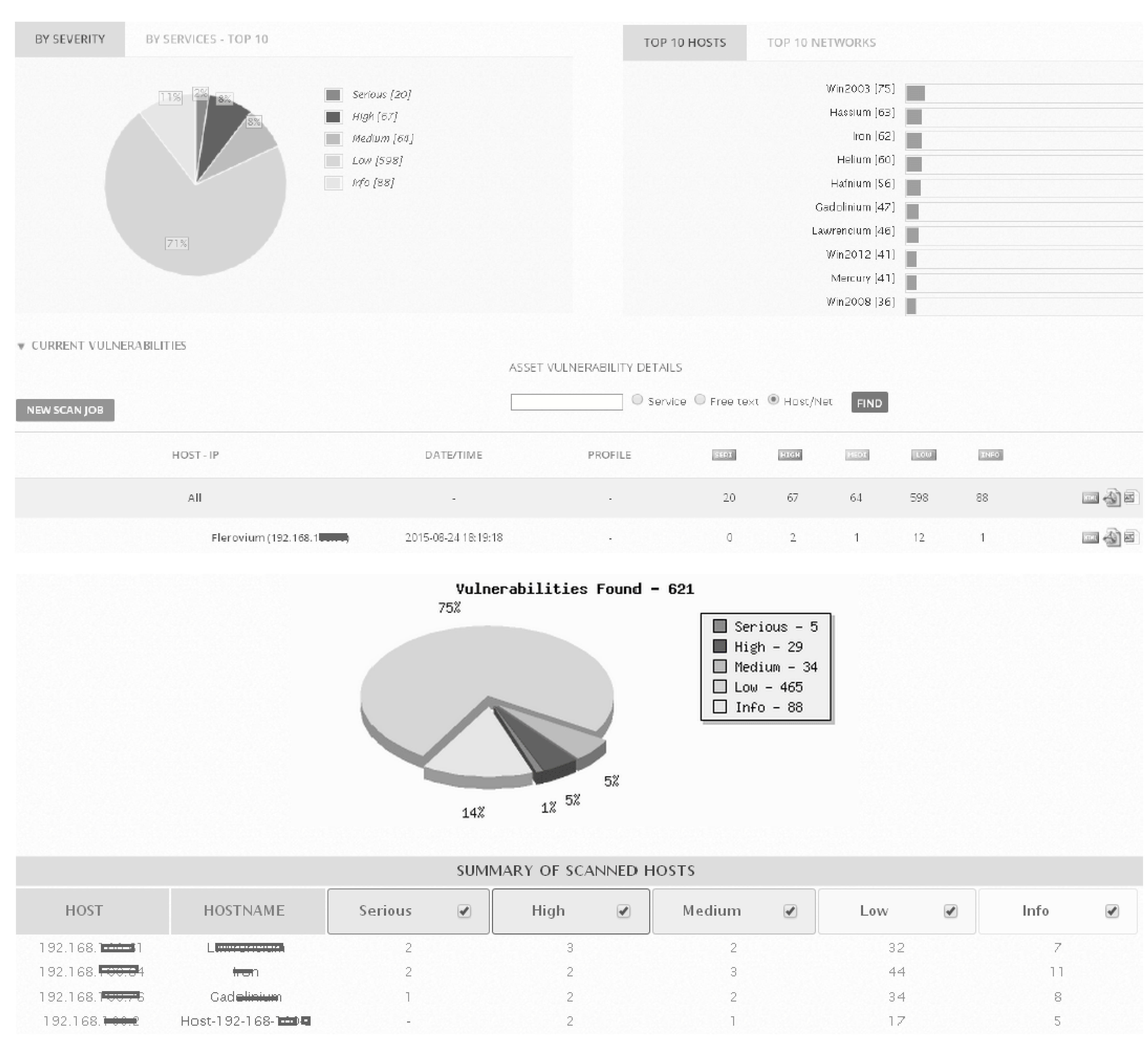


图 9-82　可视化展示漏洞扫描结果

如果要了解详情，点击某个 IP 的漏洞数目，即可显示出类似下面的一段结果。如图 9-83 所示。

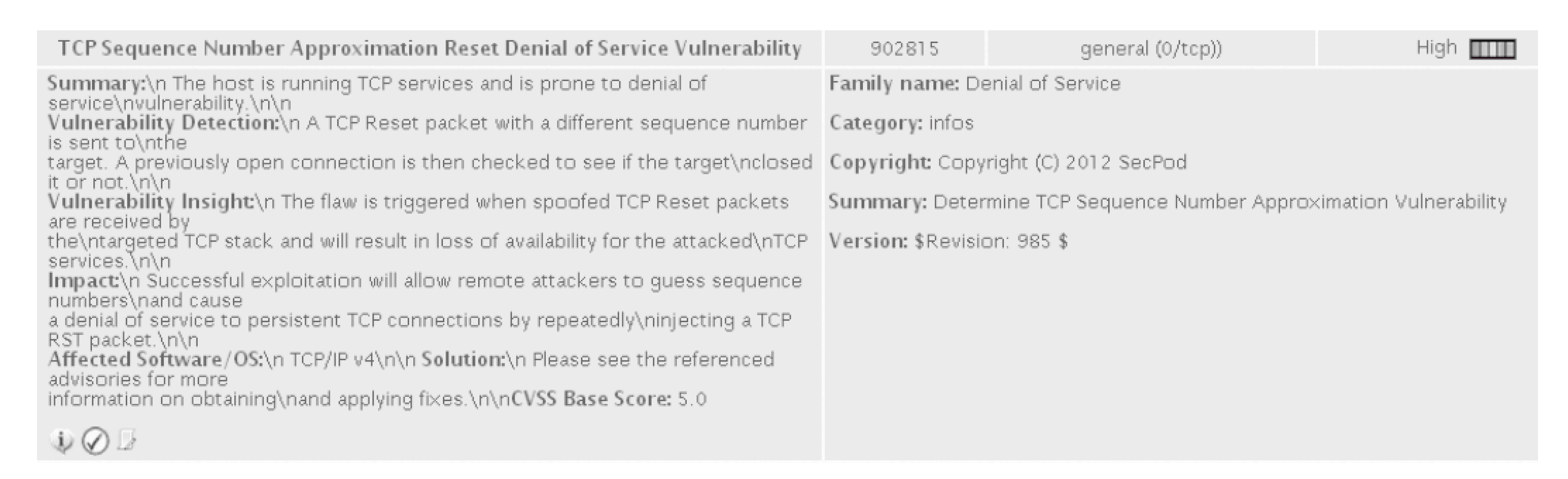


图 9-83　某个 IP 的漏洞数目

在上面的界面显示中，图标表示附加信息，表示可信，表示不可信，点击图标显示更多详细信息，尤其是 CVSS 的分值需要介绍一下，CVSS 表示 Common Vulnerability Scoring System 通用漏洞评价体系。利用该标准，可以对弱点进行评分，进而帮助我们判断修复不同弱点的优先等级，它的取值范围为 0~10，具体而言是从 0.0（不受影响的系统）到 10.0（最受影响的系统，具有很高的灾难性破坏的风险）。

（2）按服务漏洞数量排行 TOP 10

在“By Services Top 10”中能列出网络服务存在漏洞数量的排行以饼图表示，如图 9-84 所示。

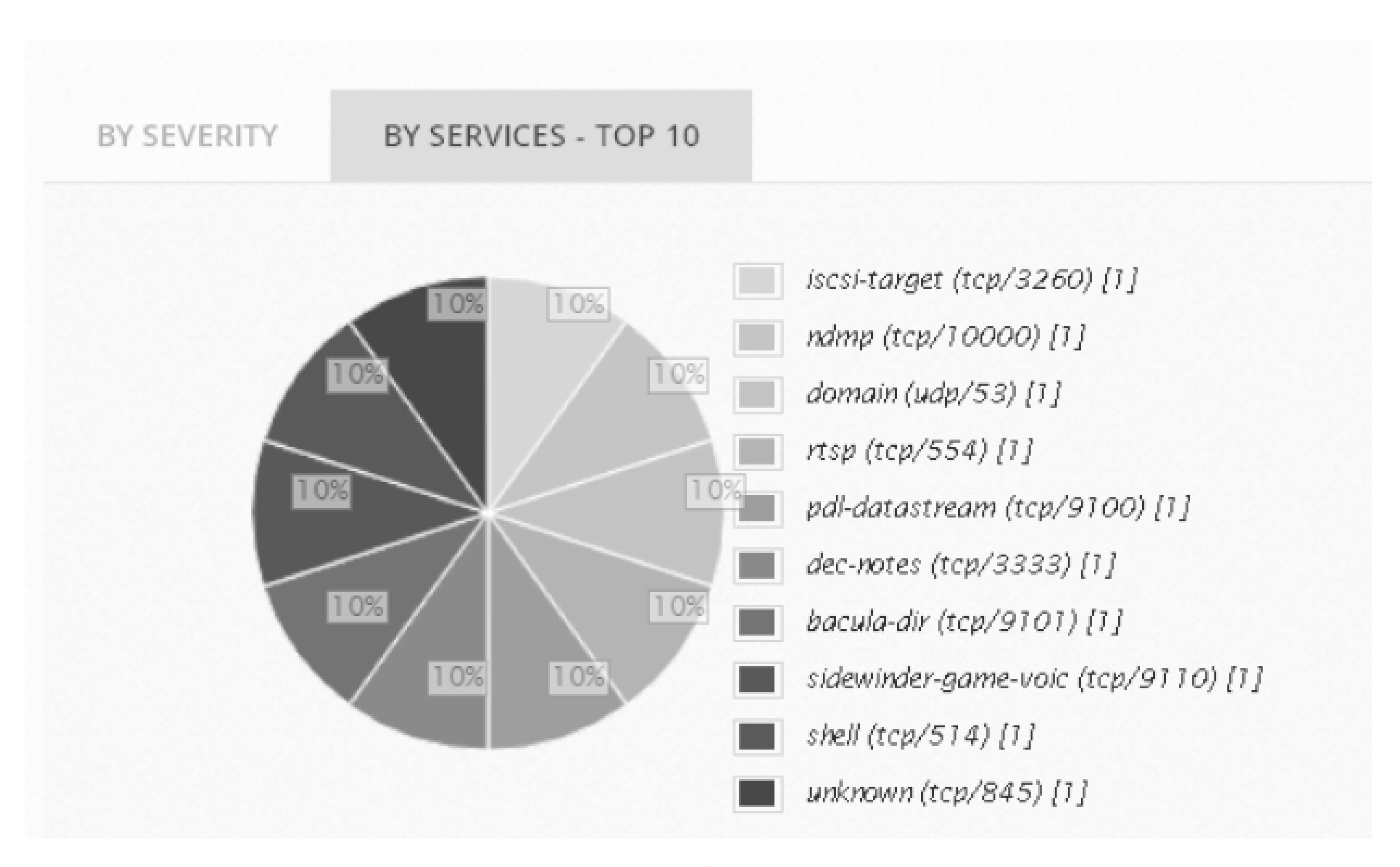


图 9-84　扫描详细结果

9.7.6　变更扫描策略

系统默认的扫描器策略非常严格，扫描内容多，过程长，如果在自己熟悉的网络环境中有时需要修改策略，则进行以下操作，这时我们需要单击 Vulnerabilities→Scan Jobs 下的“Profiles”按钮，出现如图 9-85 所示的界面。

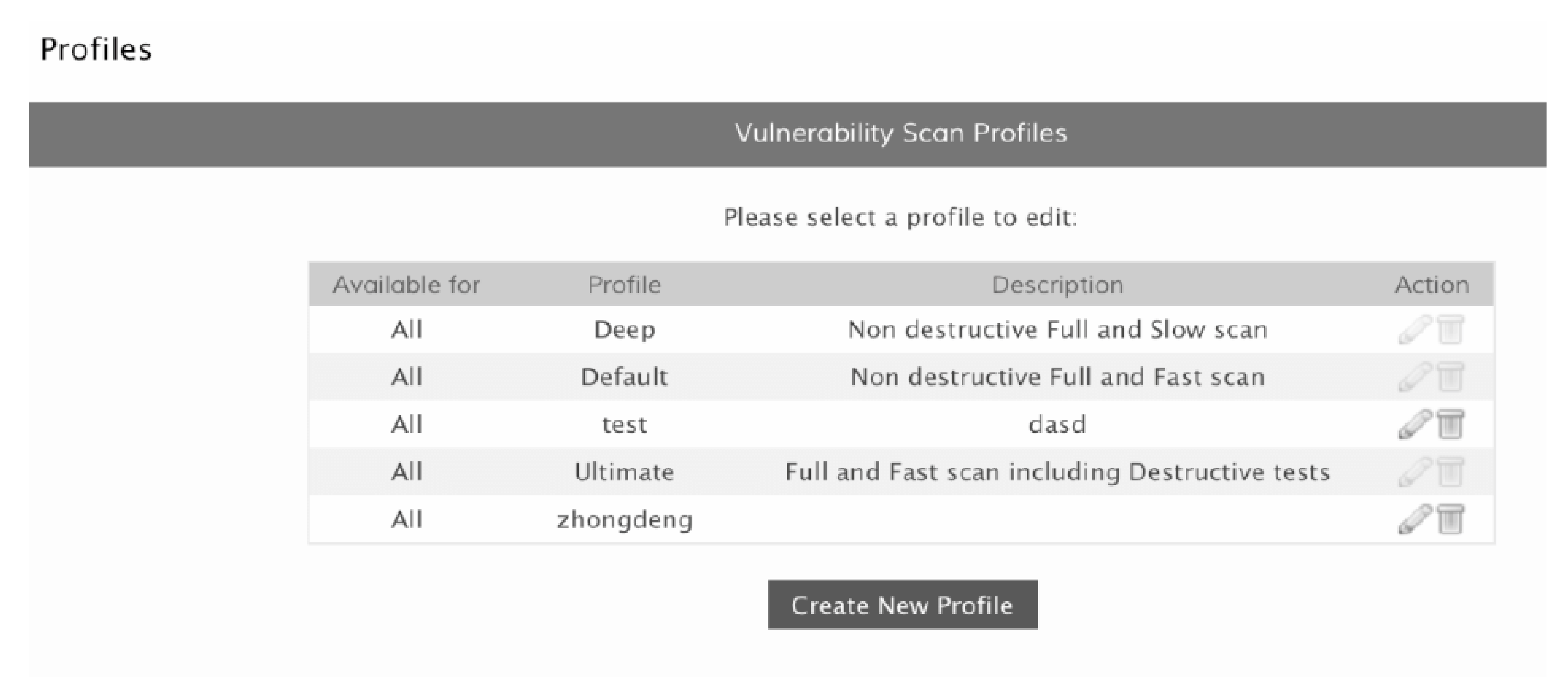


图 9-85　更改策略

单击 Creat New Profile 按钮，出现如图 9-86 所示的界面。

Profiles

New Profile

Name:	lcg
Description:	
Clone existing scan policy:	None
Make this profile available for	User: admin
Autoenable plugins option:	Autoenable by category
Set all autoenabled categories to:	Enable All

Autoenable plugins in categories:

Category	Enable All	Enable New	Disable New	Disable All	Intelligent
ATTACK	●	○	○	○	○
DENIAL	○	●	○	○	○
DESTRUCTIVE_ATTACK	○	●	○	○	○
END	○	●	○	○	○
FLOOD	●	○	○	○	○
INFOS	●	○	○	○	○
INIT	●	○	○	○	○
KILL_HOST	●	○	○	○	○
MIXED	●	○	○	○	○
SCANNER	●	○	○	○	○
SETTINGS	○	○	○	●	○

Create

图 9-86　新建扫描策略

单击创建策略，然后在 Profile 菜单中出现刚才定义的策略，如命名为“lcg”。如图 9-87 所示。

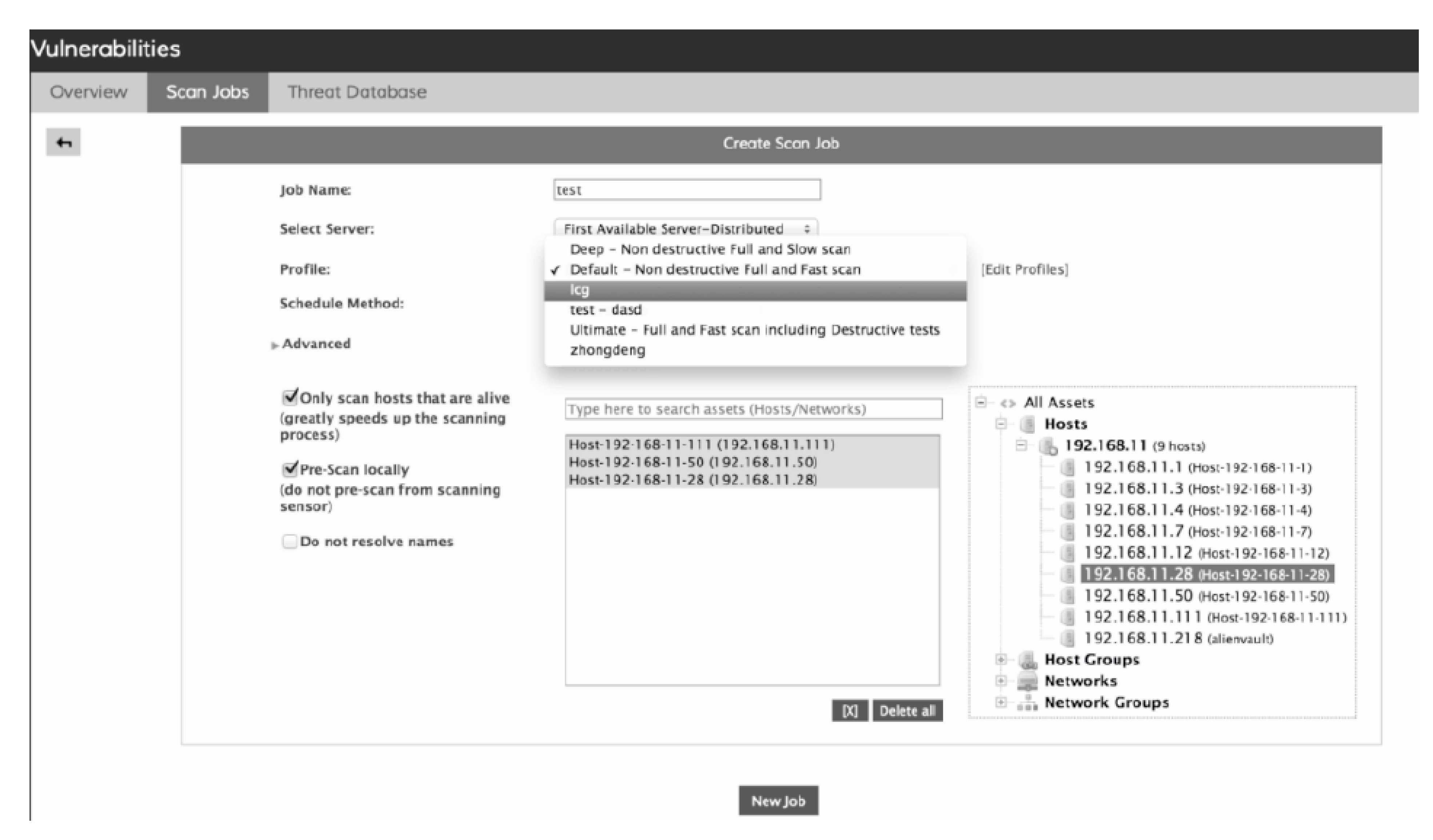


图 9-87　选择自定义策略

如图 9-88 所示，在“SCAN JOBS”中选择失败的扫描，然后单击图示右侧垃圾桶图标即可删除。

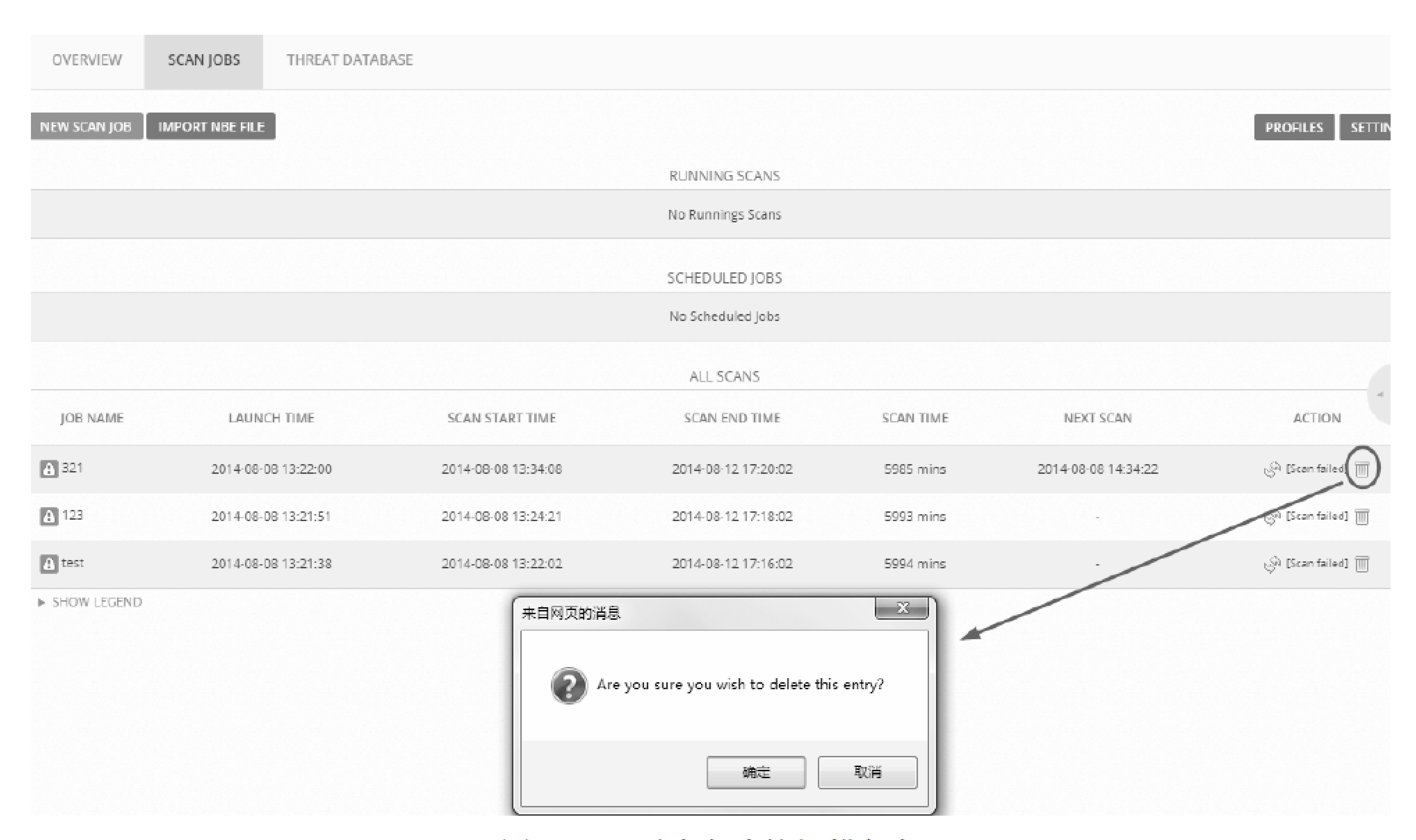


图 9-88　删除失败的扫描任务

当更改了默认策略之后，必须经过严格的测试方可通过。

9.7.7　Nmap 与 OpenVAS 的区别

有的读者容易把 Nmap 和 OpenVAS 的功能搞混淆，从范围上看，Nmap 主要用于大范围扫描，特点是速度快，但没有漏洞库，而 OpenVAS 适用于量少、多次地对局部机器进行的漏洞扫描，它默认带了 4 万多个漏洞库脚本。它们都是扫描工具，其偏重不同，Nmap 是端口扫描工具，属于轻量级工具。Nmap 的特点是结构简单速度快，用于前期侦查。

Nmap 的基本介绍和基本使用方法，互联网上提供的内容比较详细，这里给大家介绍 Nmap 可以集成的功能强大的插件，如表 9-5 所示，这些插件主要分为以下几类。

表 9-5　nmap 插件

插件名称	功 能
Auth	负责处理鉴权证书
Broadcast	探测服务打开状态
Brute	暴力破解方式，针对常见应用 http 等
Default	和使用-sC 和-A 开关一样仅提供基本扫描
Dos	进行 Dos 攻击
Exploit	利用应用程序的漏洞，进行渗透测试
External	利用第三方的数据进行 whois 解析
Fuzzer	模糊测试的脚本，发送异常包到目标主机
Malware	测试目标主机是否带有后门等信息
Version	负责增强服务于版本扫描
Vuln	负责检查目标主机是否有漏洞

使用举例：

（1）nmap --script=auth 192.168.123.*

负责处理鉴权证书的脚本，也可以作为检测部分应用弱口令。

（2）nmap --script=brute 192.168.123.*
提供暴力破解的方式可对数据库、Smb、Snmp 等服务进行简单密码的暴力猜解。

（3）nmap --script=default 192.168.123.* 或者 nmap -sC 192.168.123.*
默认的脚本扫描，主要是搜集各种应用服务的信息，收集到后，可再针对具体服务进行攻击。

（4）nmap --script=vuln 192.168.123.*
检查是否存在常见漏洞，输出内容多需要翻屏显示。

（5）nmap -n -p445 --script=broadcast 192.168.123.4
在局域网内探查更多服务打开状况。

（6）nmap --script=smb-enum-users 192.168.199.9
对 192.168.199.9 这台机器进行扫描，同时对 smb 的用户进行枚举。

（7）nmap --script=smb-brute 192.168.199.9

对 192.168.199.9 的用户名和密码进行暴力猜测。

（8）nmap --script all 192.168.0.1

使用所有脚本进行扫描。

Nmap 拥有一百多个脚本插件，详细介绍大家可以参阅 http://nmap.org/nsedoc/，在 OSSIM 中的 Nmap 工具里的插件位于/usr/share/nmap/scripts/目录，格式为*.nse。而 Openvas 是漏洞扫描工具，它主要用于对已知的主机进行漏洞扫描，结构复杂，速度慢，输出报表详细，扫描时将创建多个进程，需要消耗 CPU 资源，所以对于大规模扫描时速度慢。

OSSIM 系统在进行漏洞扫描时，首先通过端口扫描工具 Nmap，判定目标存活性以及目标系统所开放的服务和端口，然后根据服务类型与目标进行多次的数据交互，根据返回信息的特征和漏洞数据库进行匹配，来判断目标是否具有某种漏洞，最后给出详细的报表。

9.7.8 分布式漏洞扫描

在单个 OSSIM 系统中的漏洞扫描系统（Openvas、Nessus 等）采用集中式，它由中心控制节点或服务器负责进行漏洞扫描并存放结果，这种方式最严重的缺点就是当同时扫描的目标主机较多（例如有效主机大于 25 台）时服务器端会成为信息处理的瓶颈，从而导致整个系统性能下降，严重时远程控制端无法连接服务器。在升级时需要在每个 Sensor 上更新 OpenVAS 漏洞库。

如果有多个网段，大量主机需要漏洞扫描时，应分别在每个监控网段部署一个探测器进行漏洞扫描，然后将扫描结果汇总到主服务器端，进行分析并集中存储。这样可以将扫描的压力分摊到各个探测器上。我们在 Web 界面上也能同时看到两个 Sensor 的存在。分布扫描拓扑如图 9-89 所示。

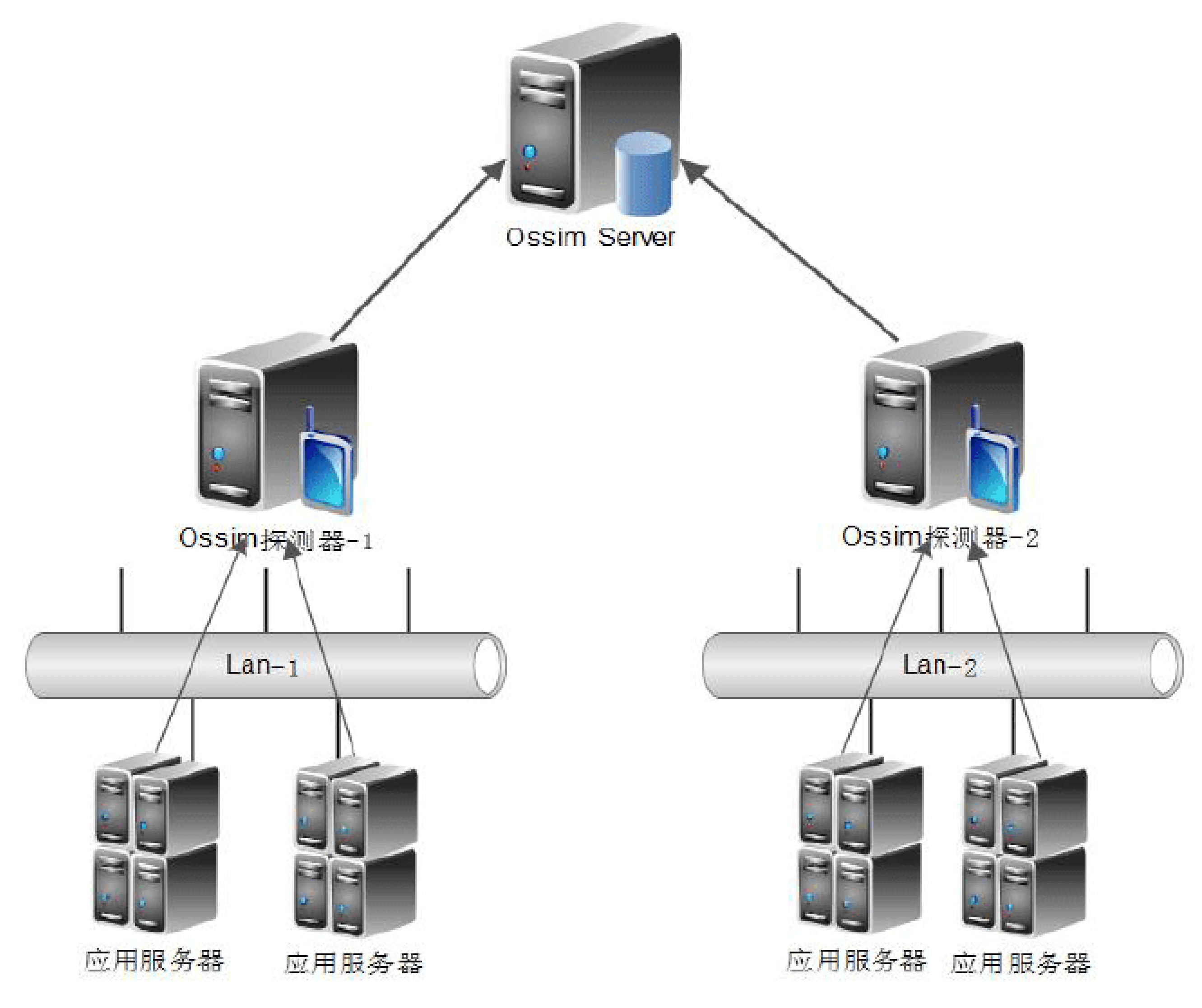


图 9-89　分布式扫描拓扑

在扫描前要选择好扫描器，OSSIM 4.8 版路径为 Configuration→Administration→Main→Vulnerability Scanner 下拉列表中选择 OpenVASManager，如图 9-90 所示。

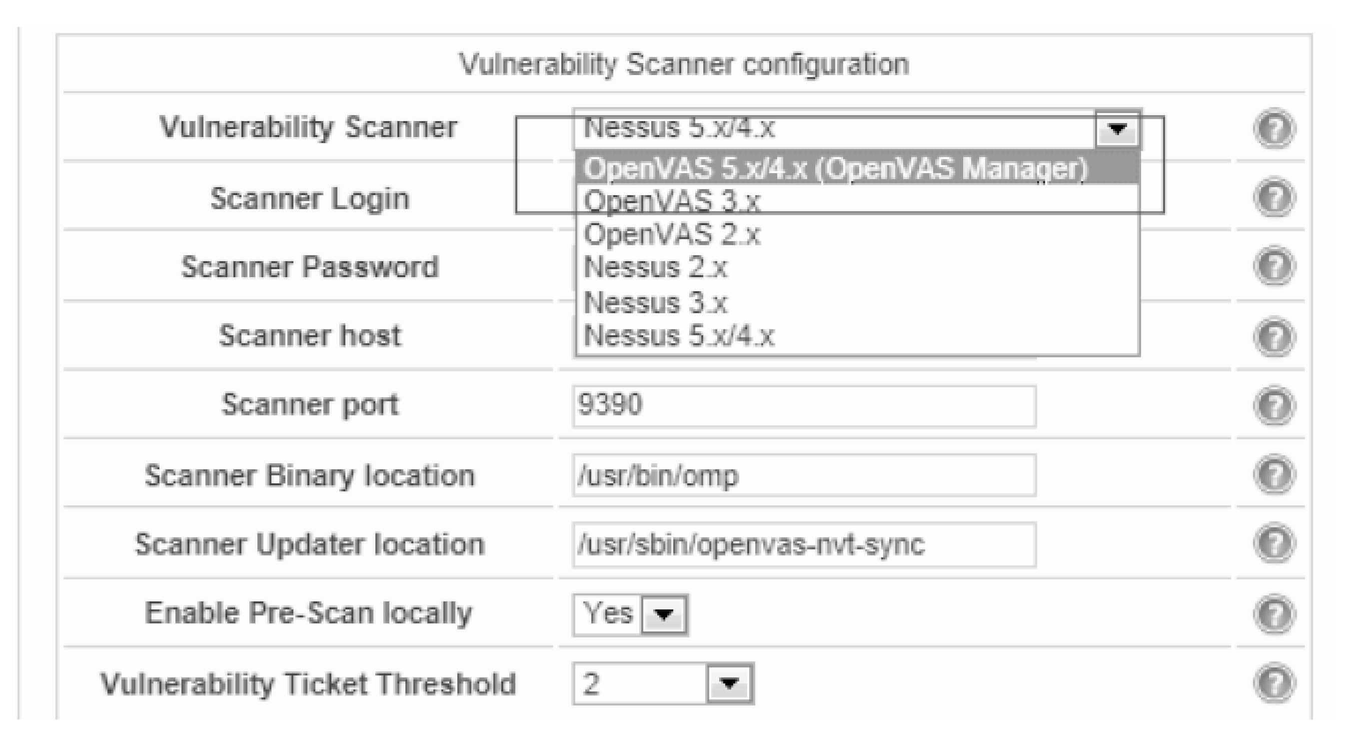


图 9-90　选择 Openvas 版本

在使用 Vulnerability 扫描时，应先将扫描主机通过 Asset 扫一遍，然后从 Host 列表中选择单台主机，不建议从一个网段开始扫描，扫描进程一旦开始就不要终止，除非整个机器死锁。

扫描结束后，我们可以在首页面板和 Analysis→Vulnerabilities 中看到结果。

9.7.9　设置扫描用户凭证

在漏洞扫描时包括直接扫描（例如 X-Scan、Nessus 等），还有带凭证的扫描测试，这里说的凭证是指访问 Openvas 服务时所需要用户名或密码。然而很多攻击者在拥有一定权限后，它们的目标在于将权限提升为超级用户，从而可执行特殊命令，因此如果赋予测试进程以目标系统的角色，将能够检测出更多漏洞，这就是带凭证测试的初衷。如图 9-91 所示的 Settings 按钮，用于检测 root 用户凭据。

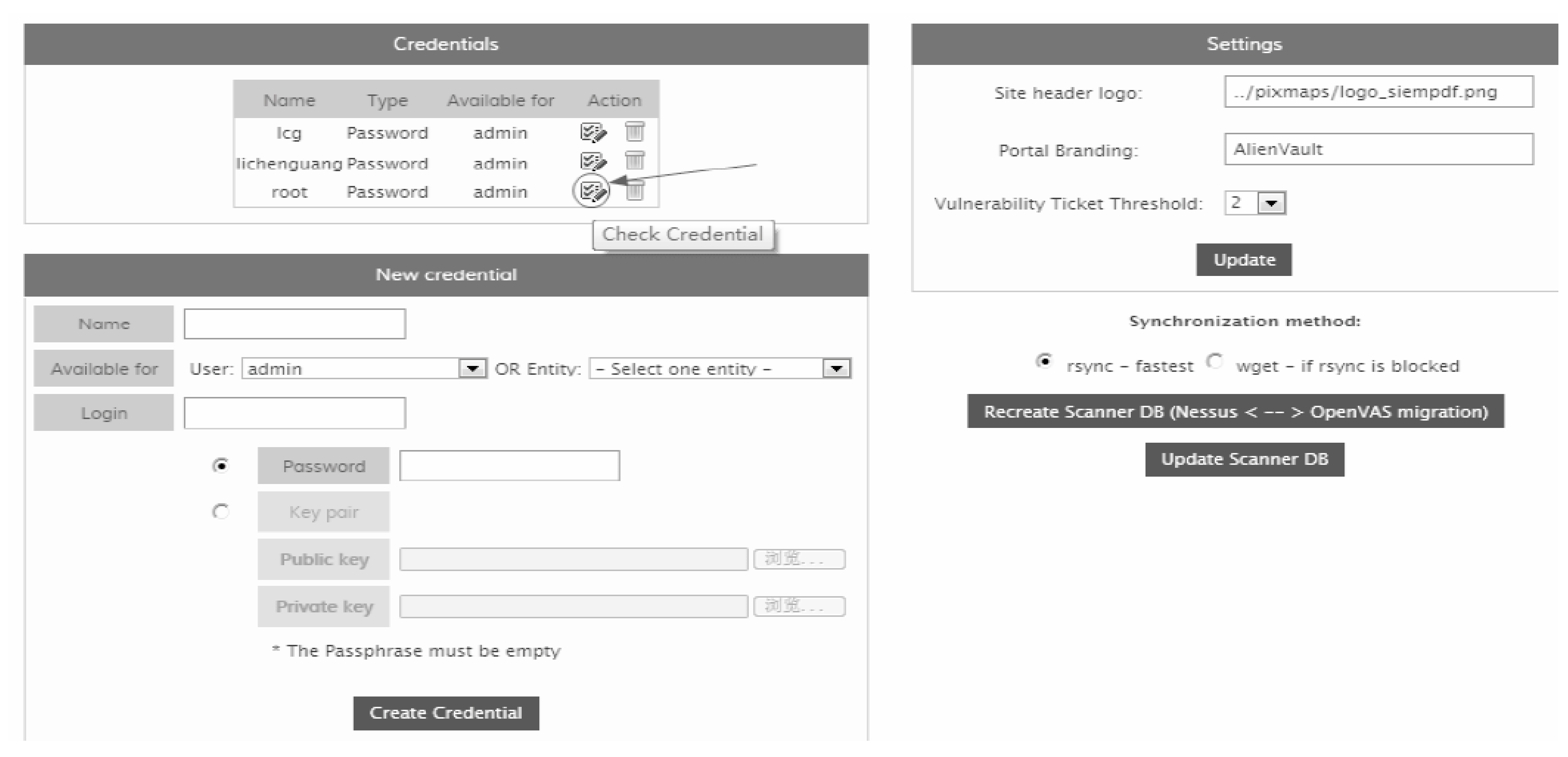


图 9-91　设置凭证

9.7.10 扫描频率

对网络资源进行漏洞扫描，如果使用不当，往往好心办坏事，例如一些用户将漏洞扫描器当作“黑客工具”滥用，在识别漏洞的过程中，它会向目标发送大量数据包，有时候甚至造成拒绝服务，无论什么漏洞扫描器会采用模拟攻击的形式，对目标可能存在的已知安全漏洞进行逐项检查，滥用所造成的影响和真实攻击造成的损失相似。所以大家在使用时一定要注意规避风险。

那么扫描周期为多少合适呢？在 PCI 2.0 中的 R.11.2 中要求至少每季度对内外网扫描一次。其实面对这一问题并没有最佳答案，好比“多久该锻炼一次”一样，尤其涉及漏洞扫描、渗透测试这种复杂的工作，要面对太多的变量，你问 10 个人可能得到 10 种不同的答案，但有一点大家需要明白，我们通过漏洞扫描需要解决什么问题？我们的目标是要找出漏洞，最大限度地减少业务系统因为系统漏洞所造成的风险，有鉴于此，漏洞扫描的次数，在保证系统稳定的情况下，越多越有利于保持安全风险的可控，那到底是每天一次?每周一次?还是每月一次?这就要看受控网络中设备的规模。

在使用漏洞扫描工具时，需要注意，对于初次使用一般是手动运行，不要使用计划任务将其设置成自动运行状态，待环境熟悉了（比如对服务器网络设备都相当了解之后），再选择计划任务自动执行。另外还要注意，漏洞扫描的并发的线程数、数据包间隔时间、扫描对象总数等，这些项应该能够调整，以便使网络的影响降到最低。图 9-92 就是 OpenVAS 在对某服务器进行漏洞扫描时，出现在 SIEM 控制台的报警。

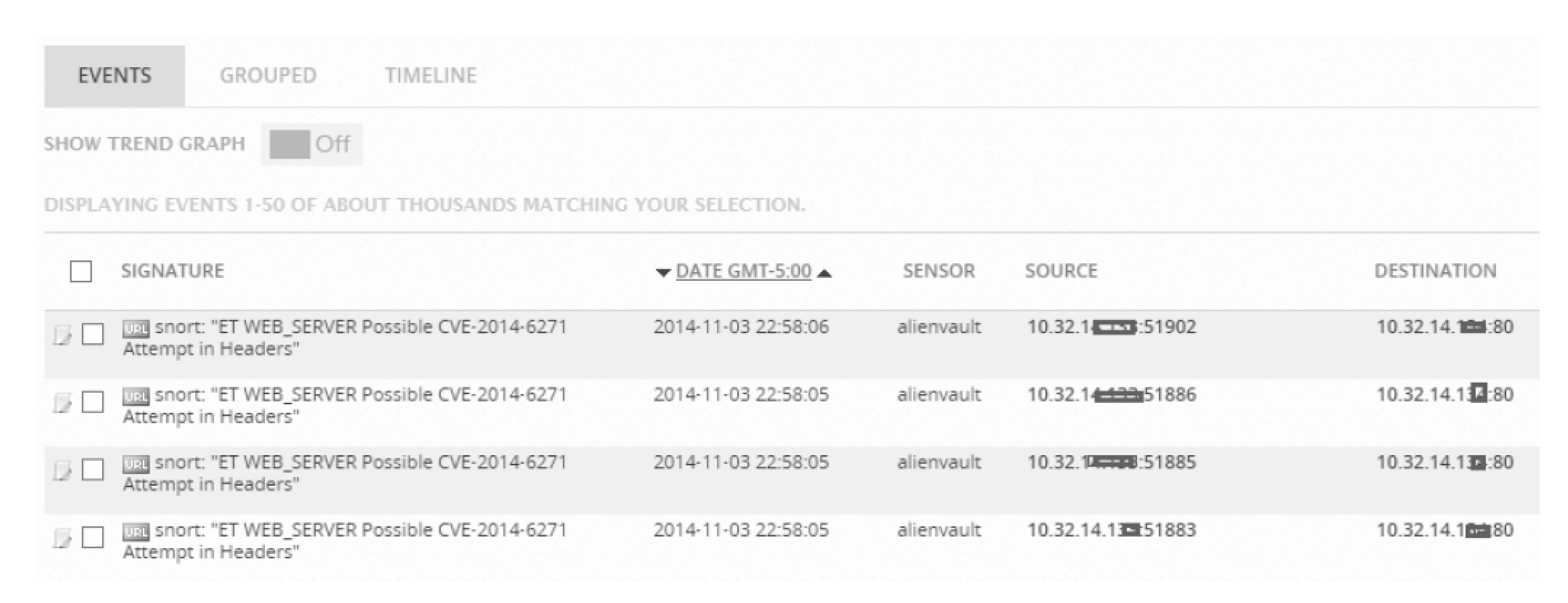


图 9-92 用 OpenVAS 漏洞扫描时触发 Sensor 上的 IDS 所呈现的事件报警

9.7.11 漏洞扫描超时问题

通常对一台主机进行漏洞扫描需要花费 8~10 分钟，而对一个网段数目不多的主机扫描通常需要数小时，根据所打开服务类型的不同，扫描时间不等，但如果扫描一台主机时间很短，小于 3 分钟或时间很长（例如大于 350 分钟），这种扫描任务应立即终止，如图 9-93 所示。

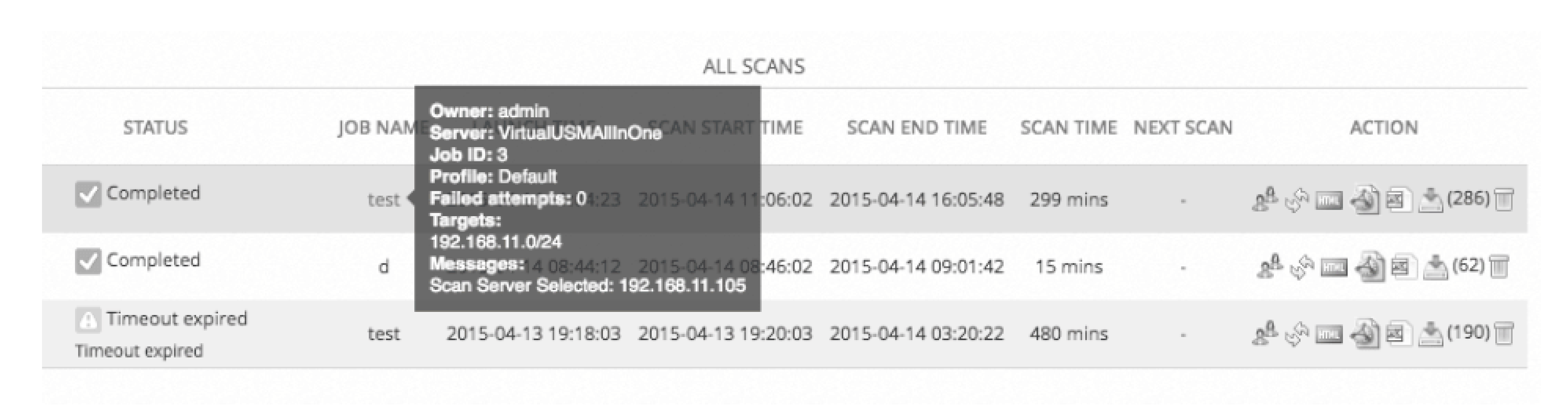


图 9-93　对一个网段进行扫描所花费的时间

9.8 OpenVAS 扫描故障排除

无论采用独立安装 Openvas 还是使用 OSSIM 集成的 OpenVAS，在使用过程中都会遇到各种问题，当然前者遇到的各种安装和编译的问题会更多，本节列举了 OSSIM 常见的使用故障并给出解决方法。

9.8.1 常见 OpenVAS 故障三则

OpenVAS 故障描述如下：

（1）修改 Root 口令导致 Openvas 无法启动。

由于用户修改了 root 账号口令，同样会导致 OpenVAS 无法启动，我们看看/etc/ossim/server/config.xml 配置文件，口令就在该文件中。如果修改了 root 口令之后必须使用 ossim-reconfig。

（2）使用 alienvault-update 命令对系统升级之后，出现 OpenVAS 无法正常工作。

OpenVAS 的故障经常出现在升级系统之后，一旦出现此类故障，我们立刻查看/usr/sbin/openvasmd 进程是否正常启动，若没有，则重启 openvasmd。如果是数据库损坏，那么需要重构数据库，方法如下：

```
# openvasmd --backup
#/etc/init.d/openvas-manager rebuild              \\重构数据库
```

下面接着重启/etc/init.d/openvas-manager 服务。

```
# netstat -npl |grep openvas
tcp 0 0 0.0.0.0:9391 0.0.0.0:* LISTEN 26976/openvassd: wai
#openvasmd --slisten 127.0.0.1 --sport 9391     \\执行后,没有任何输出代表正常\\
# netstat -npl |grep openvas
tcp 0 0 0.0.0.0:9390 0.0.0.0:* LISTEN 708/openvasmd
tcp 0 0 0.0.0.0:9391 0.0.0.0:* LISTEN 26976/openvassd:
```

最后别忘了执行“ossim-reconfig”命令，而且以上步骤是在 Server 上操作，而不是在 Sensor。

（3）对于扫描资源池之外机器的问题。

OpenVAS 对资源池内的机器漏洞扫描正常，而自己添加的其他机器扫描会出现故障。所以我们要完成一个网段内服务器漏洞扫描，首先要将机器加入到 OSSIM 资源池内，如图 9-94 所示。

DASHBOARDS　ANALYSIS　ENVIRONMENT　REPORTS

				Purpose			ms-wbt-server
✓	192.168.11.100	Host-192-168-11-100	localhost	Router, Switch	[redacted]	IOS/12.X	telnet, http
✓	192.168.11.105	Host-192-168-11-105	-	WAP, General Purpose, Broadband Router, Storage-misc, Specialized, Media Device, Network Device	[redacted]	Linux/2.6.X	ssh, http, http-ssl
✓	192.168.11.121	Host-192-168-11-121	localhost	General Purpose	[redacted]	Windows/XP	msrpc, netbios-ssn
✓	192.168.11.233	Host-192-168-11-233	localhost	General Purpose	[redacted]	NetBSD/5.X	netbios-ssn, netbios-ssn
✓	192.168.11.235	Host-192-168-11-235	localhost	General Purpose	[redacted]	Linux/2.6.X	ssh, http, http-ssl
✓	192.168.11.236	Host-192-168-11-236	localhost	General Purpose	[redacted]	Windows/XP	smtp, http, msrpc, netbios-ssn, https, microsoft-ds
✓	192.168.11.237	Host-192-168-11-237	localhost	-	[redacted]	unknown	
✓	192.168.11.238	Host-192-168-11-238	localhost	General Purpose	[redacted]	Windows/7	msrpc, netbios-ssn, netbios-ssn, rtsp
✓	192.168.11.239	Host-192-168-11-239	localhost	General Purpose	[redacted]	Windows/XP	msrpc, netbios-ssn
✓	192.168.11.240	Host-192-168-11-240	Server1	WAP, General Purpose	[redacted]	Linux/2.4.X	ssh, http, http-ssl, ntop-http, http
✓	192.168.11.245	alienvault	-	WAP, General Purpose, Broadband Router, Storage-misc, Specialized, Media Device, Network Device	[redacted]	Linux/2.6.X	ssh, ntop-http, 9390/tcp
✓	192.168.11.248	alienvault	-	General Purpose, Broadband Router, Storage-misc, Specialized, WAP, Media Device, Network Device	[redacted]	Linux/2.4.X	ssh, ntop-http, 9390/tcp
✓	192.168.11.242	Host-192-168-11-242	alienvault.alienvault	General Purpose, WAP, Media Device, Network Device	-	Linux/2.6.X	ssh, http, http-ssl, ntop-http, mysql, 9390/tcp

CLEAR SCAN RESULT　UPDATE DATABASE VALUES

图 9-94　将扫描结果加入数据库中

如图 9-94 所示，选择 UPDATE DATABASE VALUES 按钮，更新数据库。接下来，就可对其中的机器漏洞扫描。对于上面介绍的解决方法可有效处理该问题，除此之外还可执行以下命令：

- 执行“/usr/share/ossim/scripts/vulnmeter/fix_openvas_plugins.sh”
- 执行“/usr/share/ossim/scripts/vulnmeter/openvas_rebuild.sh”
- 执行“perl /usr/share/ossim/scripts/vulnmeter/updateplugins.pl”

在操作过程中还可能出现的故障现象如图 9-95 所示。

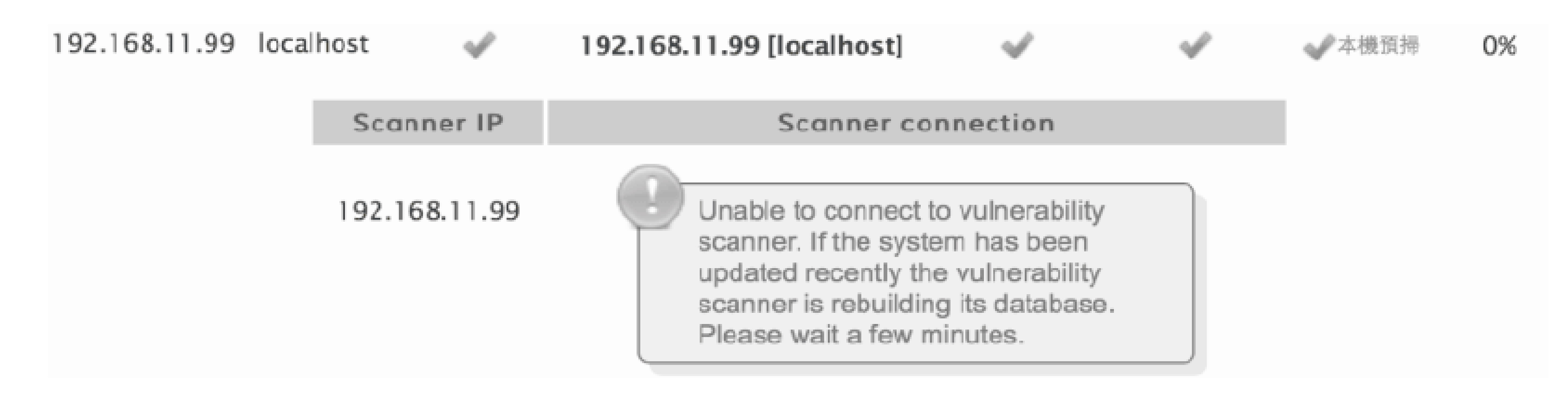


图 9-95　报告连接故障

Nessus 的守护进程的监听端口是 TCP 1241 端口，而 OpenVAS 就是 TCP 9390 端口，出现这种报错信息，主要是 9390 端口不在监听状态，我们可以查找 openvasmd 进程确认它是否启

动。如果没有启动，需要启动一下。如果直接启动也没有办法监听 9390 端口，我们应该怎么办？首先，重建数据库。

```
#/etc/init.d/openvas-manager rebuild
Rebuilding the NVT cache
```

大约 5~10 分钟后重建完成，这时重启 openvas-manager 服务。

```
#/etc/init.d/openvas-manager restart
```

（4）Sensor 与 Server 通信故障导致 OpenVAS 无法正常工作。

这种情况的典型代表是出现“Failed to authenticate”验证失败，如图 9-96 所示。

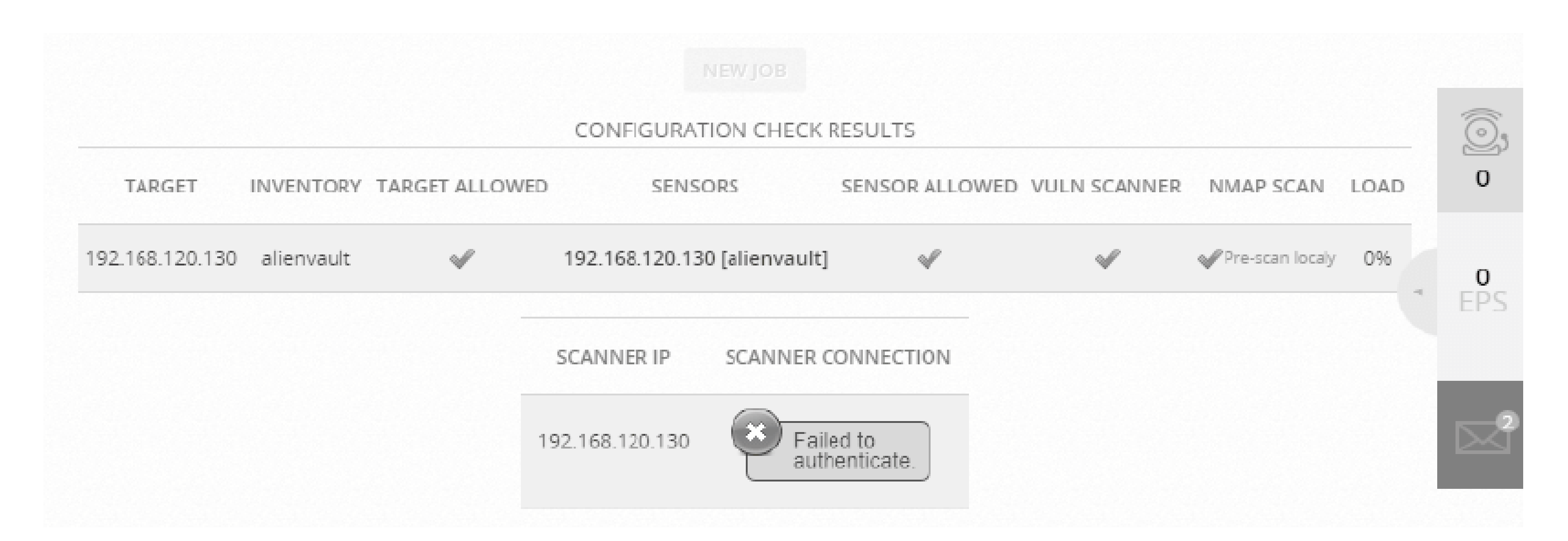


图 9-96　连接认证故障

在OSSIM 系统中使用漏洞扫描有时会出现图9-97 所示问题，图9-97 表示Sensor 无法和Server 通信截图，注意看在 Active Sensors 显示为 0，即代表异常。启动扫描进程，我们首先用 netstat -na 检查，这时发现 TCP 40001、40002 全都不在监听，查看 Sensor 状态是不正常，出现“Connection refused”表示 Sensor 无法和 OSSIM Server 通信，所以首先调整 Sensor 工作状态使其正常。

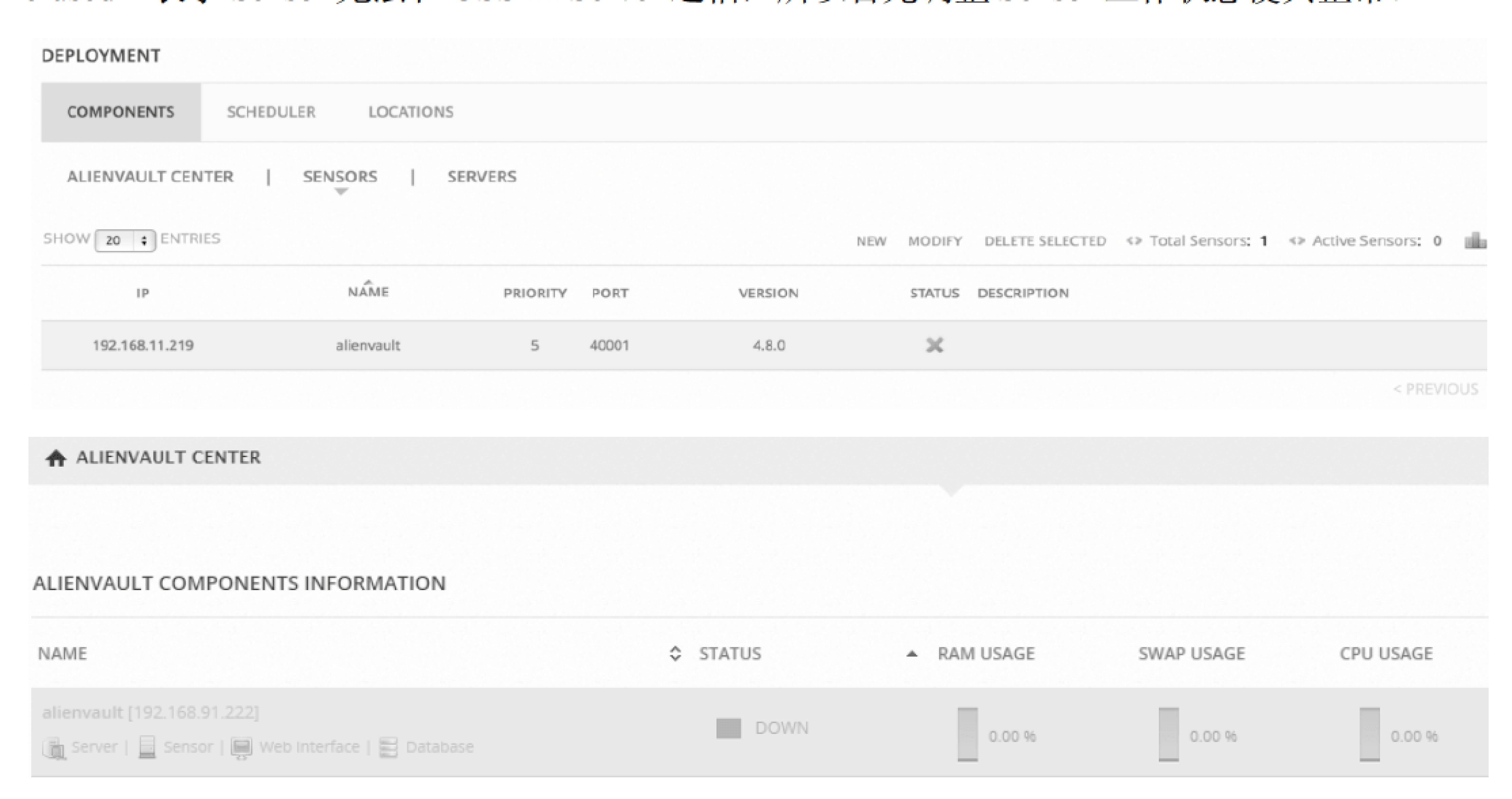


图 9-97　Sensor 通信故障

扫描时间很短是另一种扫描错误，这时会出现“Failed Service temporarily down”的提示，我们需要重构漏洞库，并重启 openvasmd-manage 服务，如图 9-98 所示。

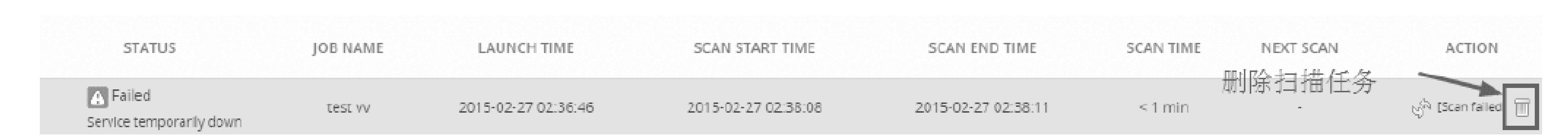


图 9-98　扫描失败

这时发现一个现象，TCP 9390 和 9391 端口都不在监听状态，与此同时，启动 Openvas-scanner 服务出现报错，如图 9-99 所示。

```
alienvault:~# /etc/init.d/openvas-manager start
Starting OpenVAS Manager: ERROR.
alienvault:~# /etc/init.d/openvas-scanner start
Starting OpenVAS Scanner: ERROR.
alienvault:~#
```

图 9-99　Openvas-scanner 启动报错

在这种故障情况下，伴随出现如下报错日志，下一小节会接着对这种故障进行详细分析。

```
alienvault:~# tail -f /var/log/openvas/openvasmd.log
lib  serv:WARNING:2014-07-03 06h18.00 utc :25307:   Failed to gnutls_bye:
GnuTLS internal error
lib  serv:WARNING:2014-07-03 06h18.00 utc :25307:   Failed to gnutls_bye:
Error in the push function
```

9.8.2　OpenVAS 故障分析

OpenVAS 扫描过程分为两个阶段，首先是端口扫描阶段，在该阶段 OpenVAS 对扫描范围内所有目标判断是否在线，然后扫描端口；接下来是探测阶段，在该阶段 OpenVAS 需要确定目标操作系统类别和开发服务器和应用类别，最后再调用相应漏洞插件对其进行深度扫描。在 OpenVAS 服务器端，可以分为 5 个模块，其工作流程如图 9-100 所示。

- 主控模块：负责控制调用各个子模块，它根据客户端配置信息调用相应的扫描数据库脚本。
- 认证模块：对客户端提交的用户名和密码进行认证校验，主要是对用户名和密码进行管理。
- 日志模块：记录日志文件，它将用户登录信息，扫描信息以及特征库更新信息写入日志文件。
- 扫描模块：根据用户的相关设置，扫描模块调用扫描数据库组装好相应数据包，发送到目标操作系统或应用，与漏洞数据库中的漏洞特征进行比较分析，以判断所选漏洞是否存在。
- 加密模块：负责对服务器和客户端之间进行通信加密，确保数据传输安全。

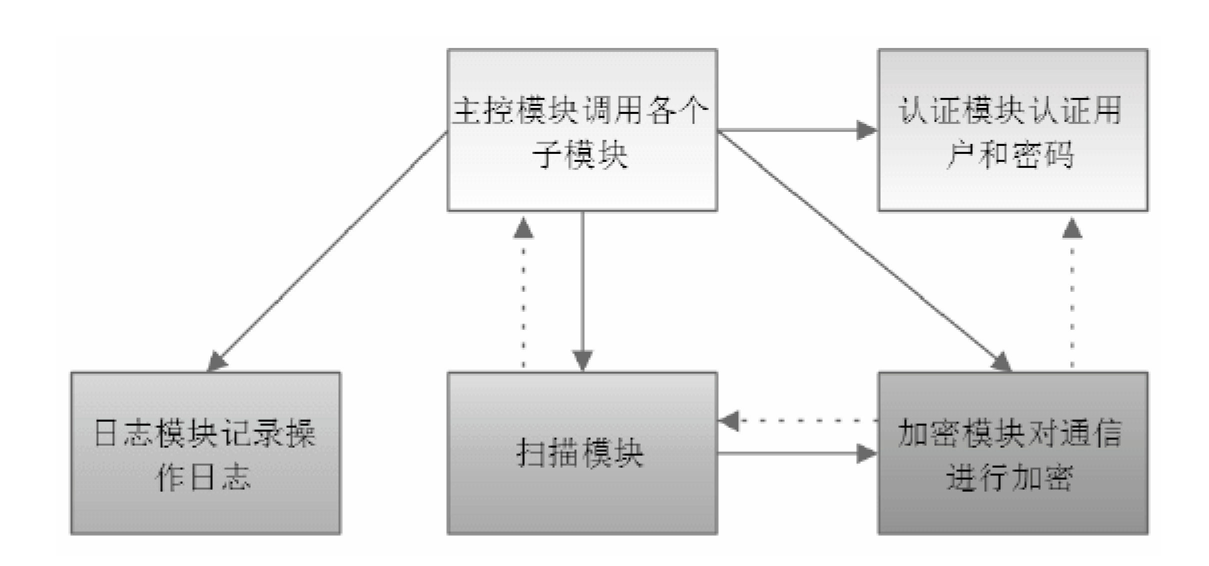


图 9-100　模块间的工作流程

OpenVAS 这种扫描漏洞库中，内部含有一些攻击性代码，因此扫描时会包含一些类似黑客攻击的扫描行为，并且 OpenVAS 服务器可以远程启动和控制，因此用户必须对扫描服务器组件的安全加密进行设置。首先由 openvas mkcert 创建证书（为 OpenVAS 安装 SSL 证书，存储在/var/lib/openvas/CA/），在启用了加密模式后，要想正常使用 OpenVAS，还必须为系统指定用户（用户名是 OSSIM），以及一次性口令（这里是用星号表示），该口令只在用户第一次登录的时候使用，一旦 OSSIM 用户登录成功，客户端就向服务器端发送公开密钥，通过这个密钥唯一标识用户，所以下次用户自动登录时就无须输入口令。

1. 故障排除方法一

（1）查看证书，如图 9-101 所示操作。

```
alienvault:/var/lib/openvas/mgr# openvas-mkcert-client  -n om -i
Generating RSA private key, 1024 bit long modulus
...........................++++++
........................++++++
e is 65537 (0x10001)
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:State or Province Name (full name) [Some-State]:Locality Name (eg,
 city) []:Organization Name (eg, company) [Internet Widgits Pty Ltd]:Organizational Unit Name (eg, s
ection) []:Common Name (eg, your name or your server's hostname) []:Email Address []:Using configura
tion from /tmp/openvas-mkcert-client.31786/stdC.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
localityName          :PRINTABLE:'Berlin'
commonName            :PRINTABLE:'om'
Certificate is to be certified until Jul  3 06:51:08 2015 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
User om added to OpenVAS.
```

图 9-101　查看 Openvas 证书

```
alienvault:/var/lib/openvas/mgr# openvasmd -backup
```

（2）停止 openvas-manager、openvas-scanner、openvas-administrator 三个服务。

（3）删除 tasks.db 数据库文件。

```
#rm /var/lib/openvas/mgr/tasks.db
```

（4）重建数据库。

```
# openvasmd --rebuild -v（重建数据库）
#cd /usr/share/ossim/scripts/vulnmeter
```

（5）更新。

```
#perl updateplugins.pl migrate
```

正常情况下 TCP 9390、9391 端口都处于监听状态，如图 9-102 所示，则 Sensor 处于活动状态。

```
alienvault:/usr/bin# netstat -npl|grep openvas
tcp        0      0 0.0.0.0:9390            0.0.0.0:*               LISTEN      9586/openvasmd
tcp        0      0 0.0.0.0:9391            0.0.0.0:*               LISTEN      9819/openvassd: wai
tcp        0      0 127.0.0.1:9393          0.0.0.0:*               LISTEN      4199/openvasad
alienvault:/usr/bin# _
```

图 9-102　查看 9390 等各端口状态

（6）最后，重新开始扫描目标机器。

在正常时，日志显示如下：

```
#tail -f /var/log/openvas/openvasmd.log
event auth:MESSAGE:2014-07-03 06h16.06 utc :25106: Authentication success for
user ossim (e015fbaf-7e65-4f05-adbb-2e400b96e620)
```

2. 故障排除方法二

如果还未解决问题，可以采用以下方法。

```
#openvasad -c remove_user -n ossim        \\操作效果如图9-101所示。
#openvasad -c add_user -n ossim -r  Admin
```

提示：该命令用来设定管理员账号，目的是添加一个管理员角色的 OpenVAS 登录用户，执行时会跳出要设定 admin 账号的密码的提示，如果想更改管理者账号名称，需要将该行中的小写 admin ，改成想要的账号名称。如图 9-103 所示。接着，在 Web UI 中输入新口令。

```
You have new mail in /var/mail/root
alienvault:~# openvasad -c remove_user -n ossim
ad   main:MESSAGE:22438:2014-07-28 23h48.42 EDT: User ossim has been successfully removed.
alienvault:~# openvasad -c add_user -n ossim -r Admin
Enter password:
ad   main:MESSAGE:23581:2014-07-28 23h49.07 EDT: No rules file provided, the new user will have no r
estrictions.
ad   main:MESSAGE:23581:2014-07-28 23h49.07 EDT: User ossim has been successfully created.
alienvault:~#
```

图 9-103　添加用户

下面在终端验证是否修改生效：

```
alienvault:~# openvasmd --database=/var/lib/openvas/mgr/tasks.db
--slisten=127.0.0.1 --sport=9391 --listen=127.0.0.1 --port=9390
```

如果没有任何报错信息，则代表设置成功。下面开始 Web UI 设置，分为以下几个步骤：

首先选择 Configuration→Deployment→Components，然后选择当前的主机，例如 alienvault，如图 9-104 所示。

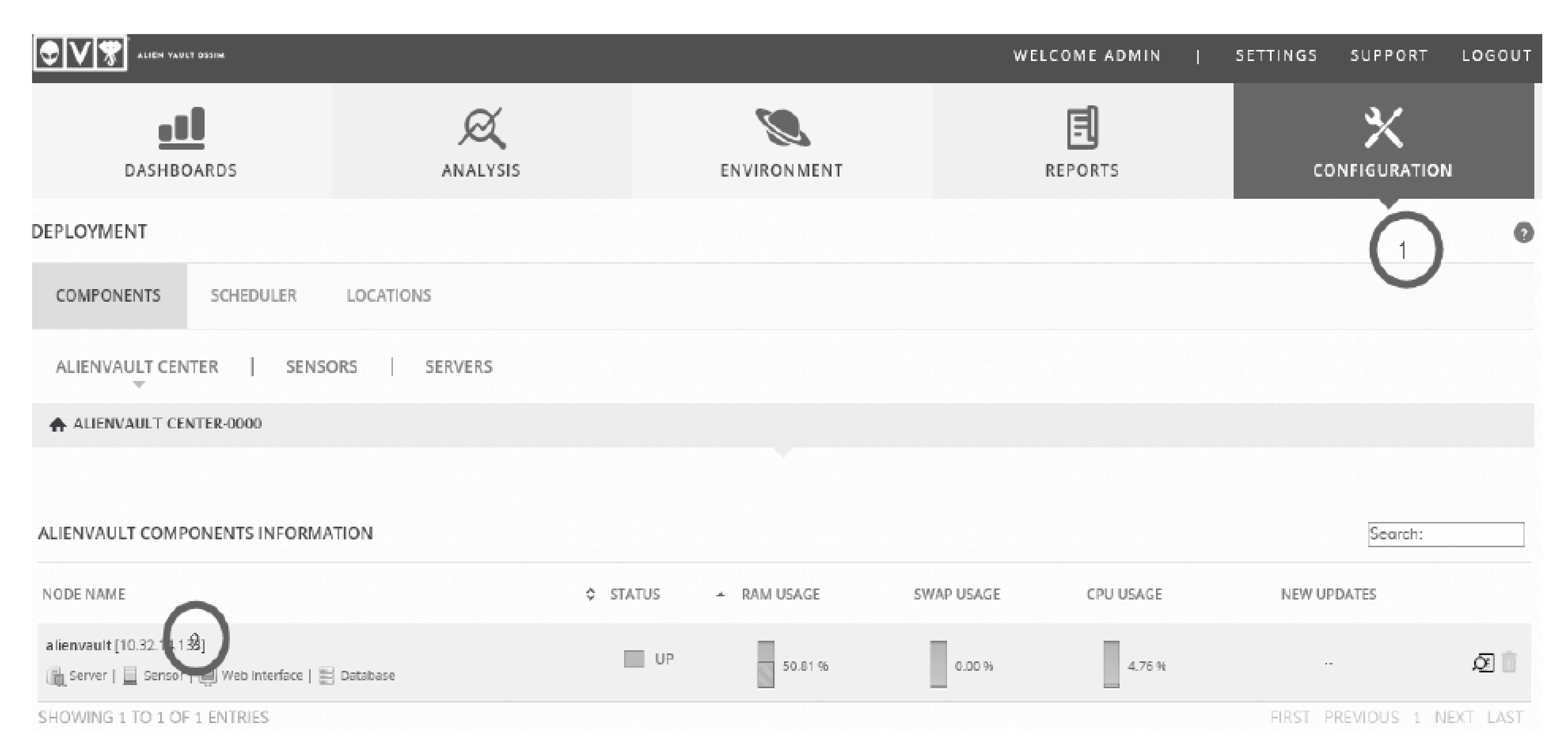


图 9-104　选择某台 OSSIM 主机

收集系统状态和内存、CPU 信息可能需要一段时间，待收集完成就会呈现带颜色的实体图标。接下来，我们选择 Sensors 按钮，选择当前 OSSIM 的 IP 地址，如图 9-105 所示。然后在弹出对话框中找到 Services 栏，输入用户 ossim 的口令，最后单击 UPDATE 按钮，如图 9-106 所示。接下来，即可以正常操作。

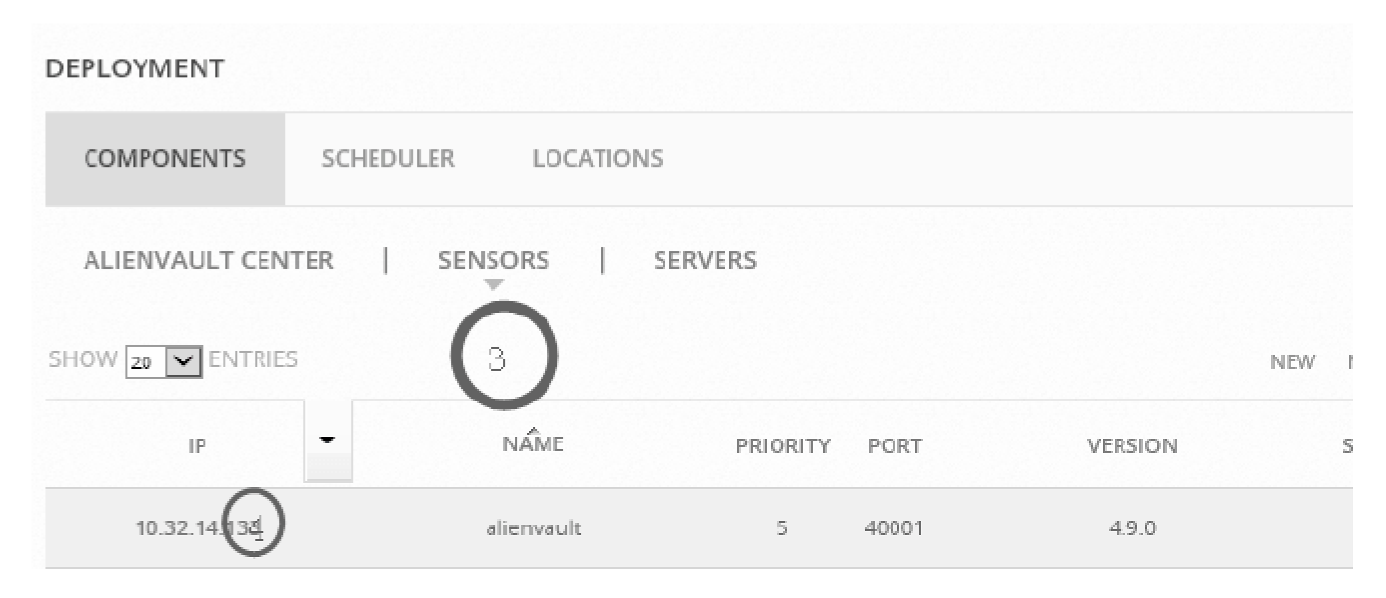


图 9-105　选择一台 OSSIM

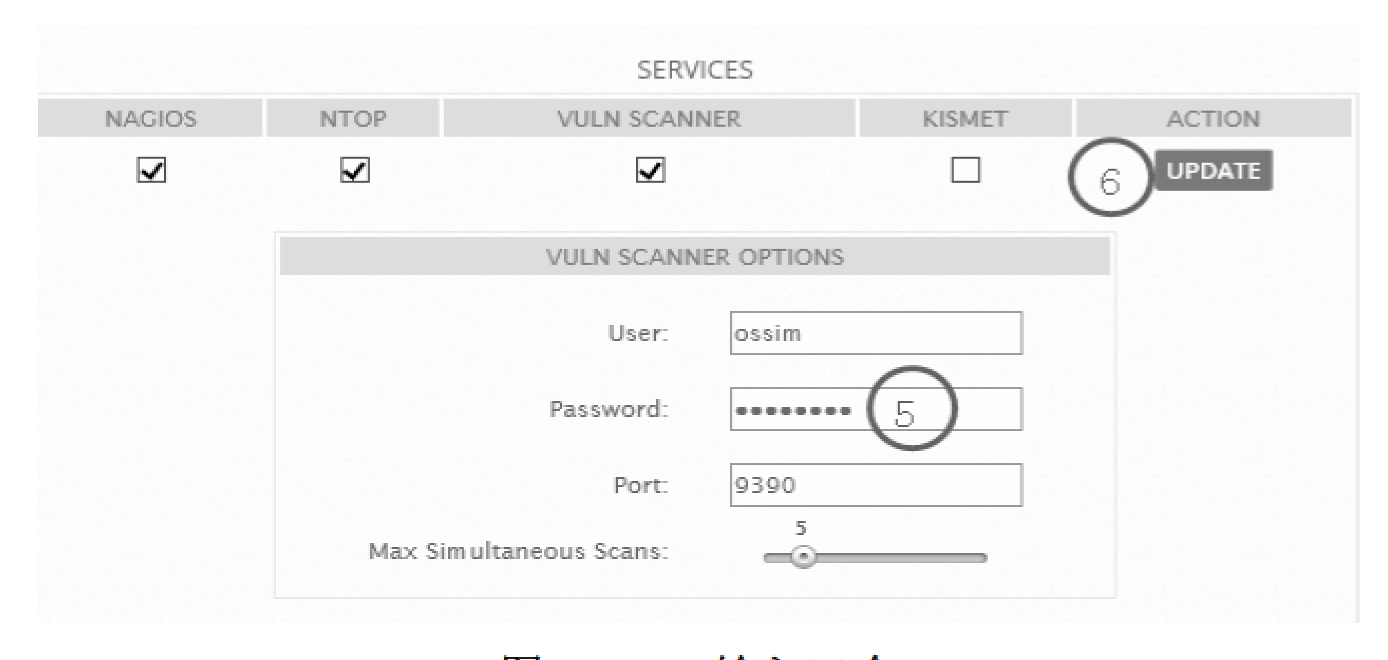


图 9-106　输入口令

9.9 配置 OSSIM 报警

9.9.1 基本操作

（1）首先登录 admin 控制台，选择 Configuration→Administration→Main，在 Tickets 选项中，选中“Mail Server Configuration Settings”选项，如图 9-107 所示，这里可以设置 SMTP 服务器地址和发送端口以及用户名和密码等信息。然后在 Ticket 选项中选择 Tickets parameters 选项，设定 Send email notification 为“Yes”，默认每天最大发送邮件数量为 15 个，可以根据自己需要修改。

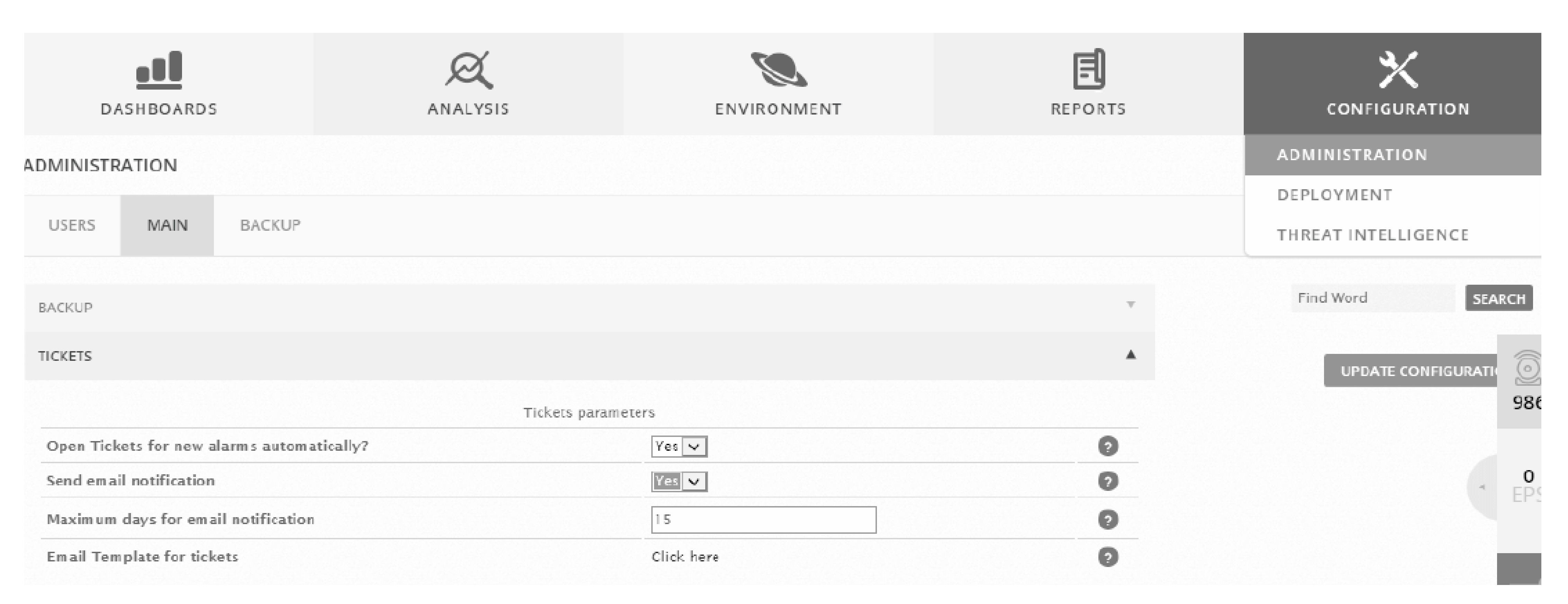


图 9-107 查看/更改 Ticktes 设置

在 Email Template for tickets 一栏还可以定制发送邮件模板，如图 9-108 所示。

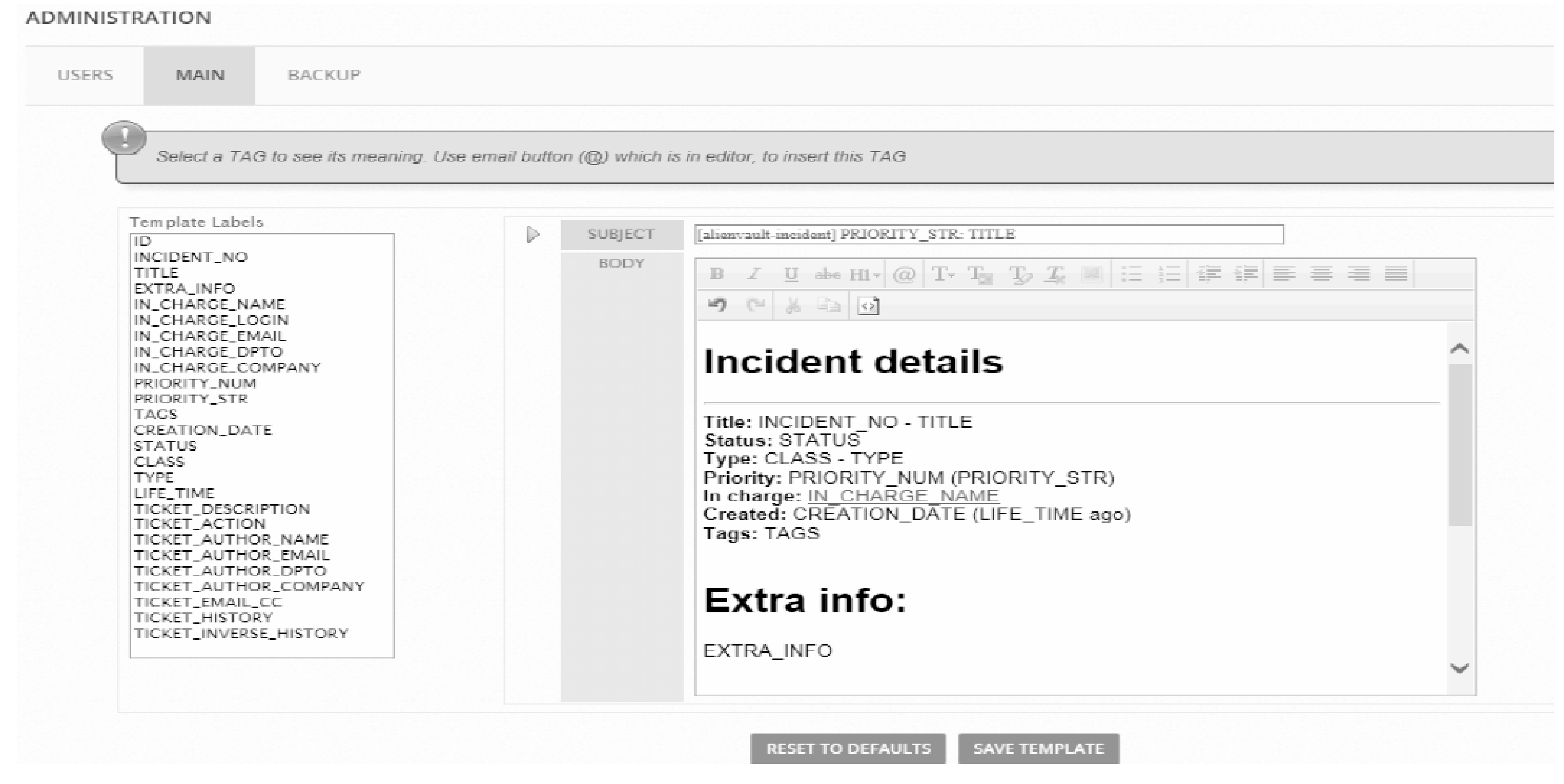


图 9-108 自定义邮件模板

（2）编辑/etc/ossim/ossim_setup.conf 配置文件，按下列格式修改，注意域名 test.com 为示例。

```
email_notify=admin@test.com
mailserver_relay=yes
mailserver_relay=mailrelay.test.com（没有域名的填写 IP 也可以）
mailserver_relay_passwd=unconfigured
mailserver_relay_port=25
mailserver_relay_user=unconfigured
```

检查/etc/postfix/main.cf 的配置是否正确。

（3）重启服务：

```
#ossim-reconfig -c -v
```

（4）测试邮件：

```
#echo "text" | mail -s "Subject" user@domain.com
#tail -f /var/log/mail.log
```

（5）在 Intelligence→Policy & Actions 菜单中创建策略和动作，如图 9-109 所示，详细操作在下面的例子中给出。

图 9-109　配置邮件报警

（6）日志测试

```
ossim:~#grep ossim@  /var/log/ossim/*
/var/log/ossim/frameworkd.log:2013-05-13 10:20:30,809 Action
[INFO]:successful response with action:Mail test@test.com
```

9.9.2　实例

下面的例子主要目的是设置发送 SSH 登录告警提示，我们知道 OSSIM 策略，如图 9-110 所示，那么系统策略分为三种，分别是：

- Default Policy Group：默认组策略；
- AV Default Policy：Alienvault 默认策略；
- Policies for events generated in server：事件产生策略。

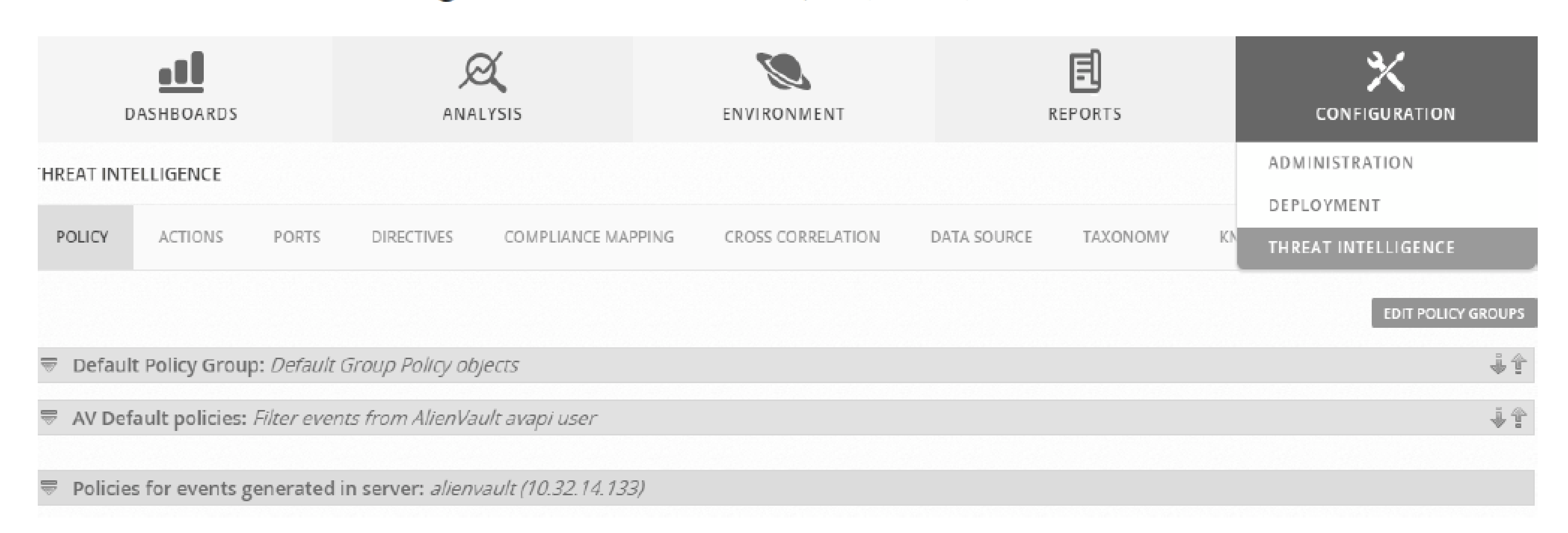


图 9-110　默认策略

当修改策略后需要重启服务，方可生效！

以下操作以 OSSIM 4.8 平台为例，首先在 Web 界面下选择 Configuration→Threat Intelligence 开始新建一个策略，如图 9-111 所示。

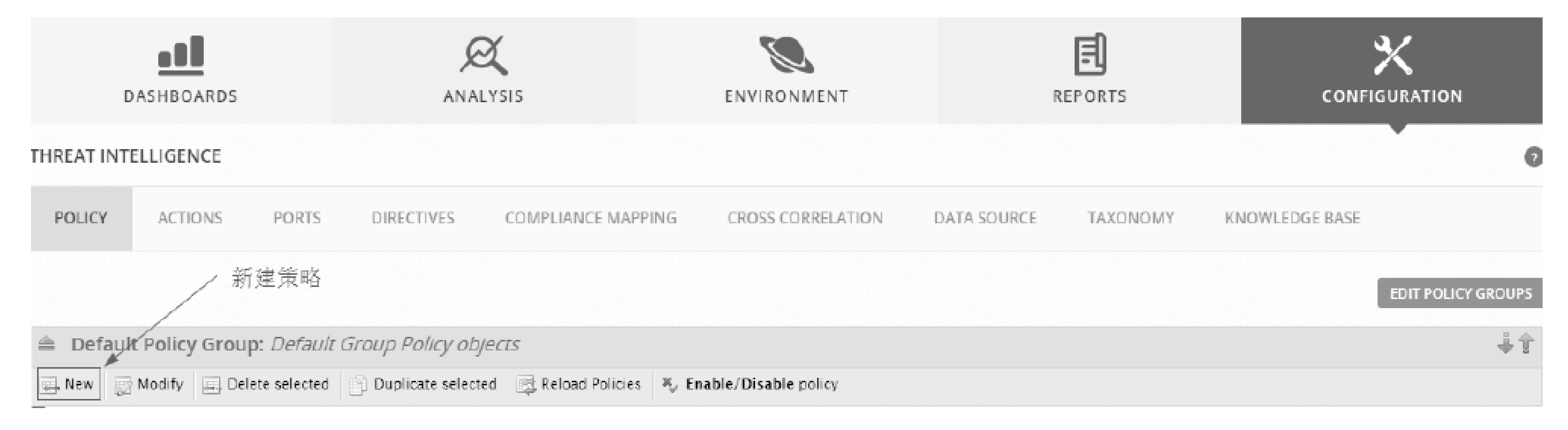


图 9-111　新建策略

一条策略包含两个部分：条件和行动，可以形象地比喻为“If That, Then This”。策略主要就是调整 Conditions（条件）和 Consequences（结果）选项的内容。网络分析师可以通过策略发现、捕获网络中各种故障。

首先，选择 SOURCE（源地址）和 DEST（目标地址），然后可选择一个或多个 DS 组或插入一个新数据源，接着选择 INSERT NEW DS GROUP 按钮，弹出如图 9-112 所示的画面。

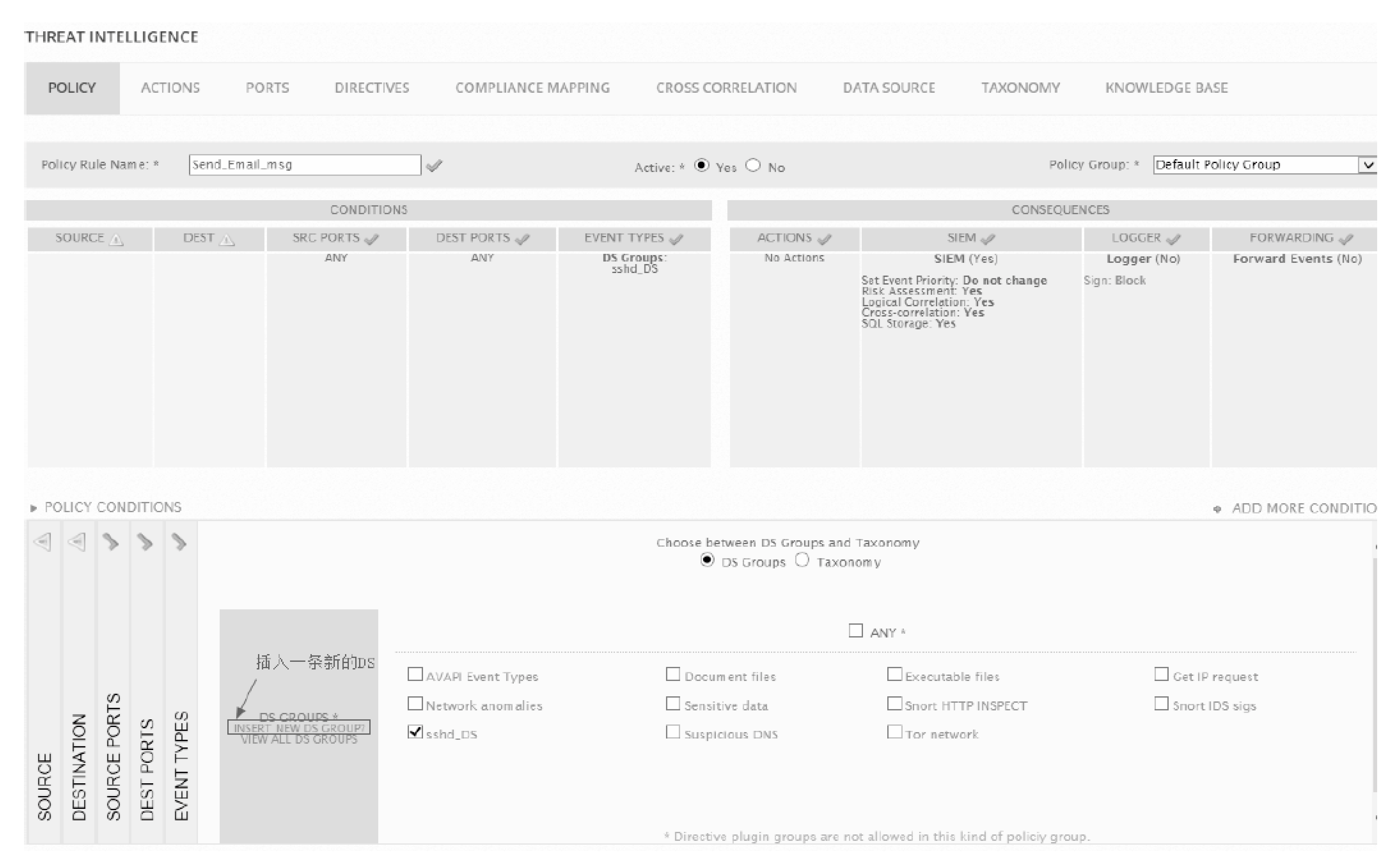


图 9-112　添加新的数据源（DS）

我们可以在列表中选择数据源，如果没有找到，还可以在右侧搜索栏键入数据源名称，例如 SSH，当选择完毕后，单击“添加数据源”按钮，如图 9-113 所示。

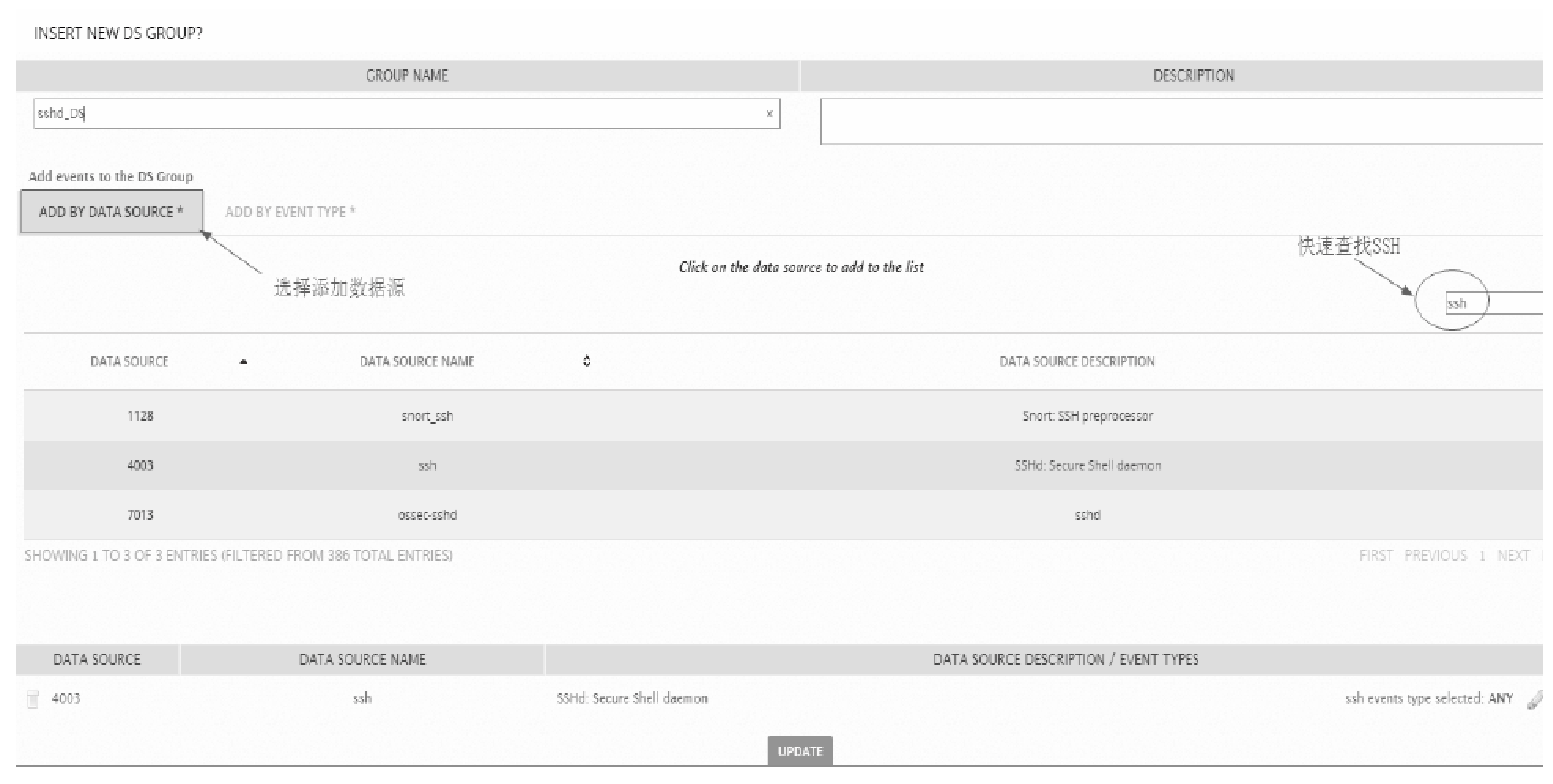


图 9-113　添加数据源

下面，我们可以选择添加动作（Actions），单击插入新动作（Insert new action）按钮，如图 9-114 所示。添加完成后下面开始设置邮件。

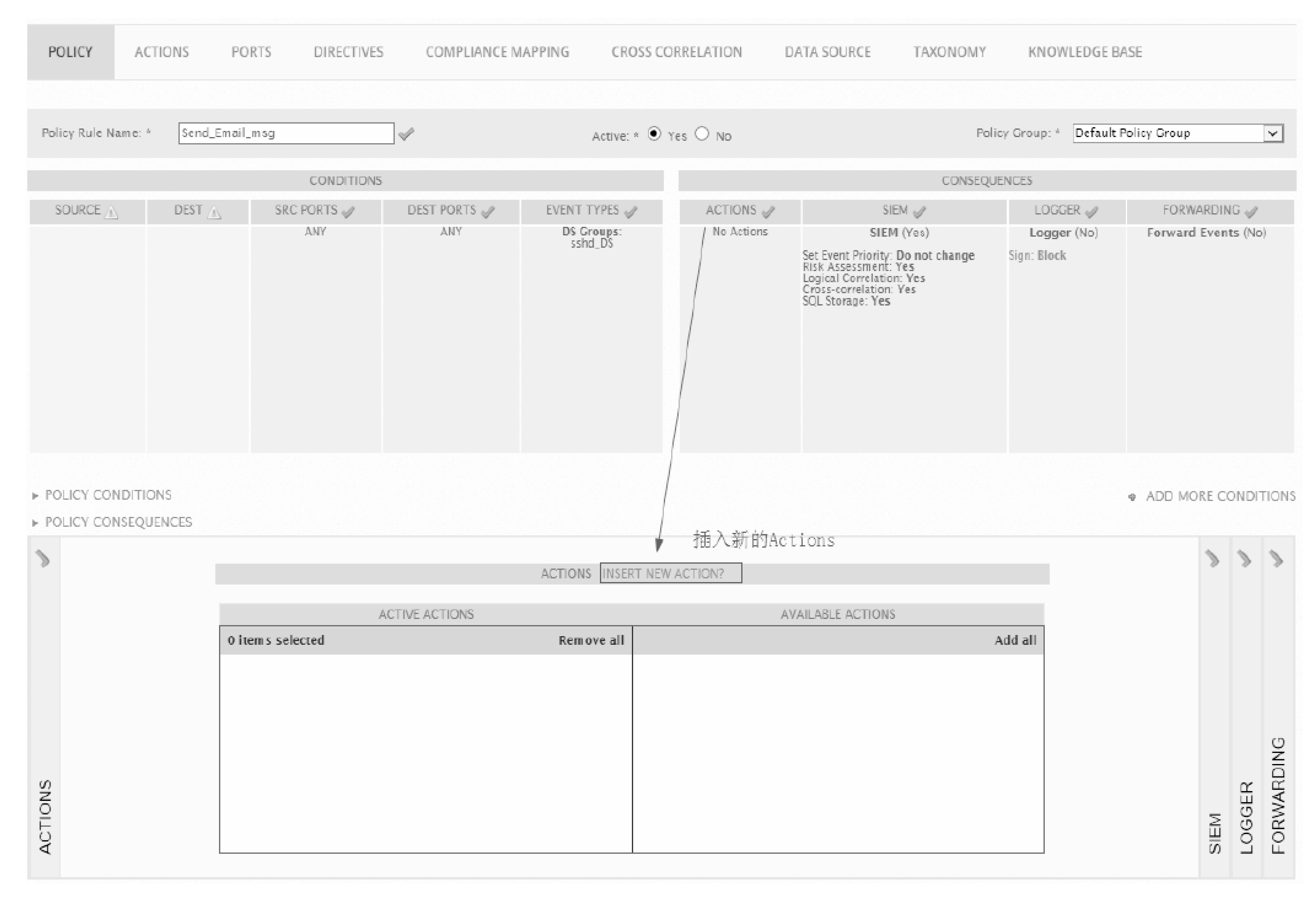


图 9-114　插入动作

最后保存以更新策略，单击 Reload Policies 按钮重新加载策略，如图 9-115 箭头所示位置，此时基本设置完毕，当然有关策略的设置方法和技巧，还不止这些，更多技巧等待大家去发现。

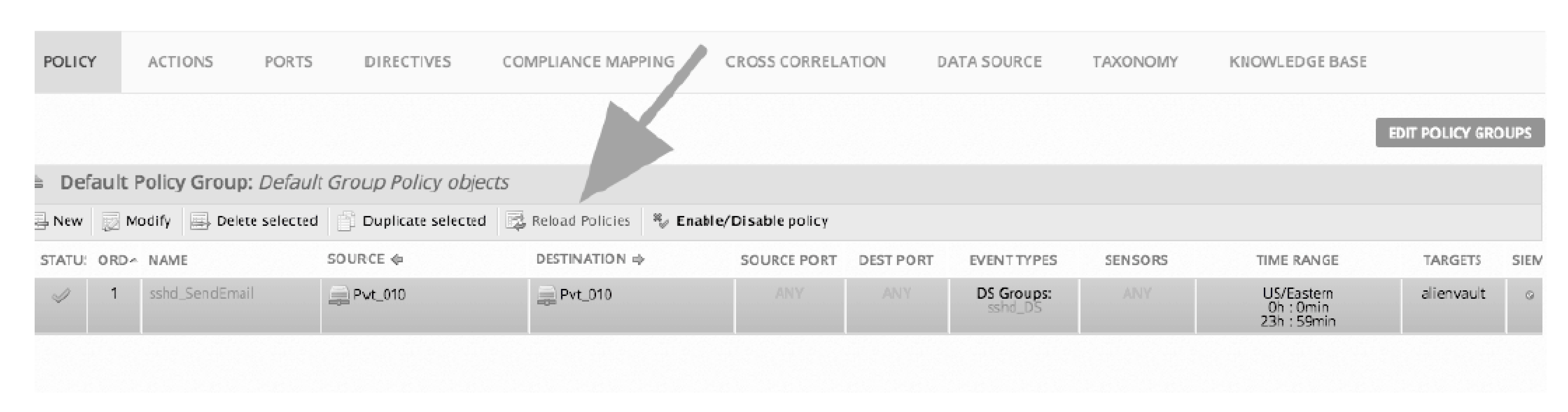


图 9-115　重新加载策略

9.10　OSSIM 在蠕虫预防中的应用

蠕虫的传播会消耗大量的网络链路可用带宽，造成网络的不稳定甚至瘫痪。我们用常规技术虽然不能彻底根除蠕虫，但我们可以采取措施将其影响尽量缓解，从而保证网络的稳定运行。

9.10.1　多维度分析功能

通过 OSSIM 提供的开源软件从多维度、多技术的视角发现蠕虫攻击，以及感知前期 APT 的攻击行为，包括对数据包的审计，进行数据包深度内容分析，异常行为分析，攻击事件分析，时间线分析以及日志分析等大数据处理。特别是通过 OSSIM 内的智能事件关联分析引擎，能够做到攻击确认、事件关联可疑发现、因果溯源的功效，如图 9-116 所示。通过时间线（Time Line）分析可以了解单位时间内蠕虫爆发频率。

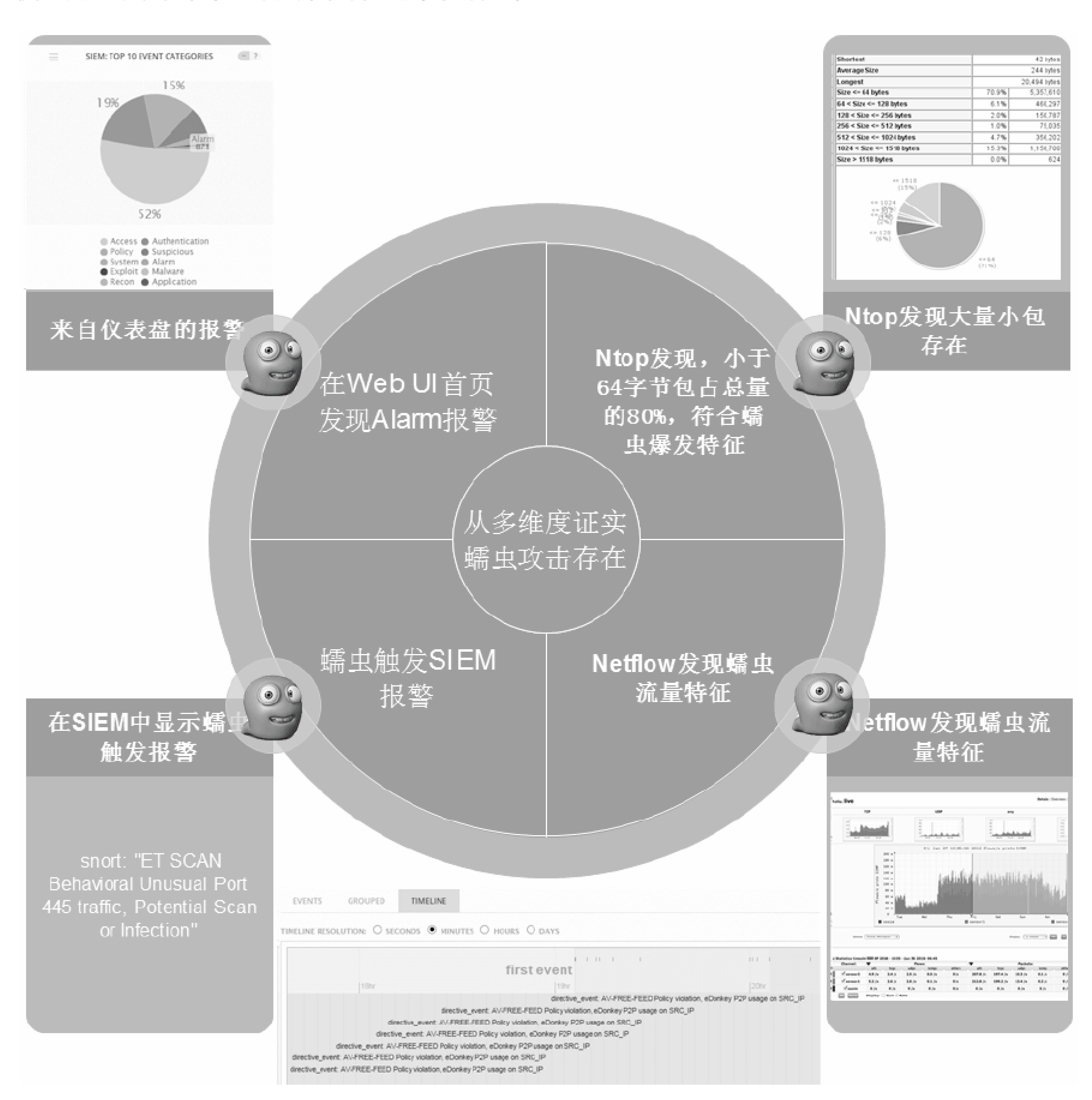


图 9-116　通过 OSSIM 多维度发现蠕虫攻击

9.10.2　发现异常流量

我们知道利用 Cisco NetFlow 所采集和输出的网络流量的统计信息，可以发现单个主机发出超出正常数量的连接请求，这种大量异常的流，往往是蠕虫爆发或网络滥用的迹象。因为蠕虫在发作时会扫描大量随机 IP 地址，来寻找可能的目标，会产生大量的 TCP、UDP 或 ICMP 流。尽管 NetFlow 不能对数据包做出深层分析，但是已经有足够的信息来发现可疑流量。NetFlow 记录非常适用于早期的蠕虫或其他网络滥用行为的检测。利用这种方法，一般在几分

钟内就能跟踪到其源头的 IP 地址、MAC 地址、所连接的交换机和端口号信息，最后将其端口关闭隔离。

9.10.3 蠕虫分析

下面再分析个实例，如果一个主机使用端口 445 连接到 5 个不同的主机上，这可能是一个正常的行为。如果它连接到 15 个主机？我们开始觉得可疑。如果它们在很短时间内，连接 100 个不同的主机呢？确认这种攻击变得更加可靠。在 OSSIM 显示面板中，可以将日志分类中发送日志最多的前 10 位列出来，以引起管理人员的注意，如图 9-117 所示。这里我们关注 Malware 类的所有日志信息，要了解图中其他各种分类，可在 Configuration→Threat Intelligence→Taxonomy→Categories 中可查看详情。

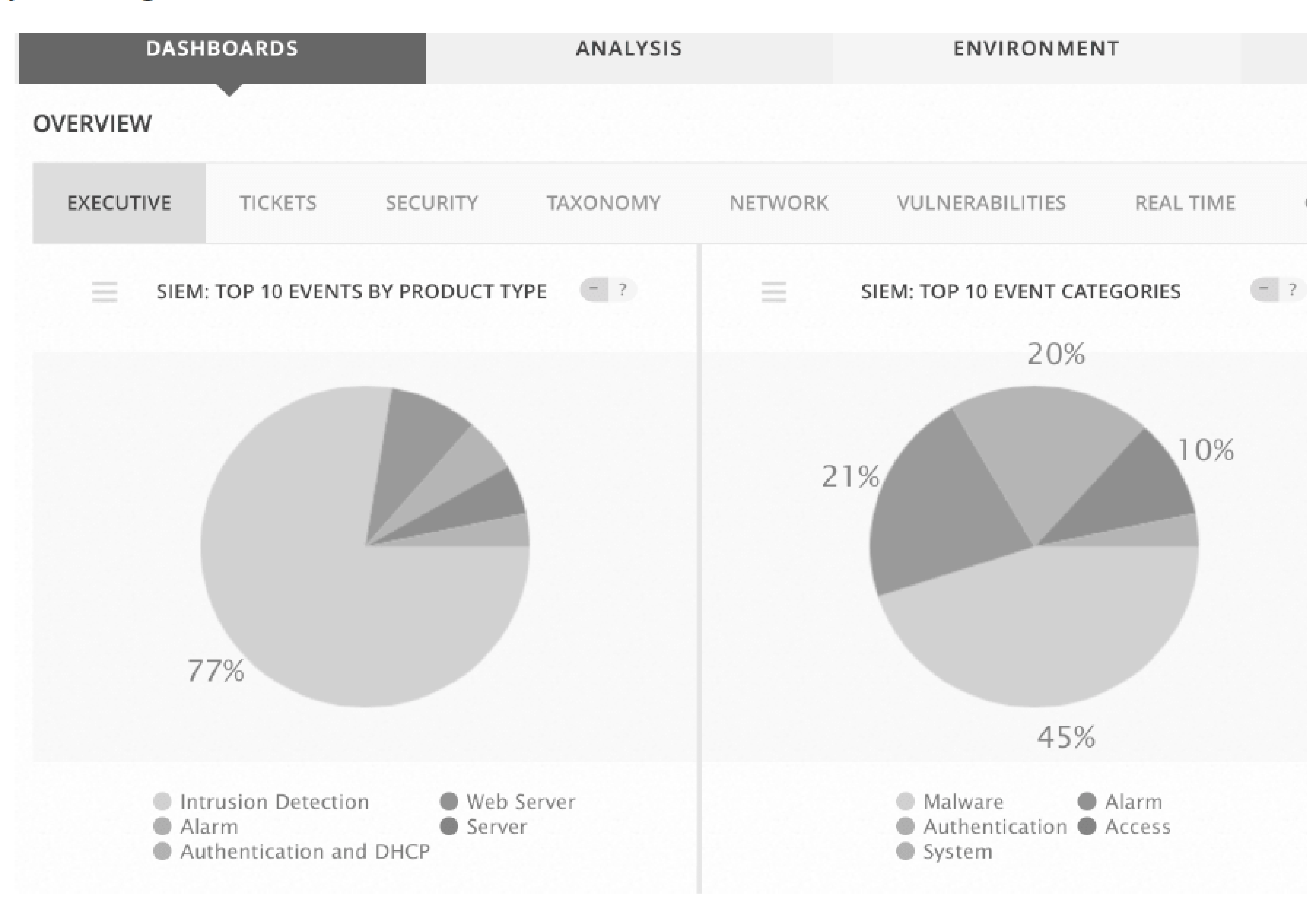


图 9-117 按日志类别分类显示 Top 10

经过观察详细日志，立即能够发现日志的风险等级都比较高。

Top10 Events by ProductType”和“Top 10 Event Cotegories”表达的含义有何不同？

这些都是基于分类的比较。而且它们之间也有共性，也就是同一类事物往往会具有更多的近似特性，如果查看插件，它们有产品型号参数，再查看插件中的 SID，它们有事件类型和子类型。就像在 OSSIM 中把报警 Alarm 这类事件又细分为 Attacks、Bruteforce、Dos、Misc、Network、Scan 等若干子类一样。

总的来说，这些只是重在用不同的方法去分析相同类型的信息，都是分类和编码日志消息含义的一种手段，这就好比在生物学中将不同生物的不同特征归属于不同界、门、纲、目、科一样。

对于扫描 Windows 主机 445 端口，这种蠕虫的攻击日志发送得非常频繁，我们可以用下面讲到的时间线分析法来分析，打开某一个日志能够看到里面记录的细节，如图 9-118 所示。

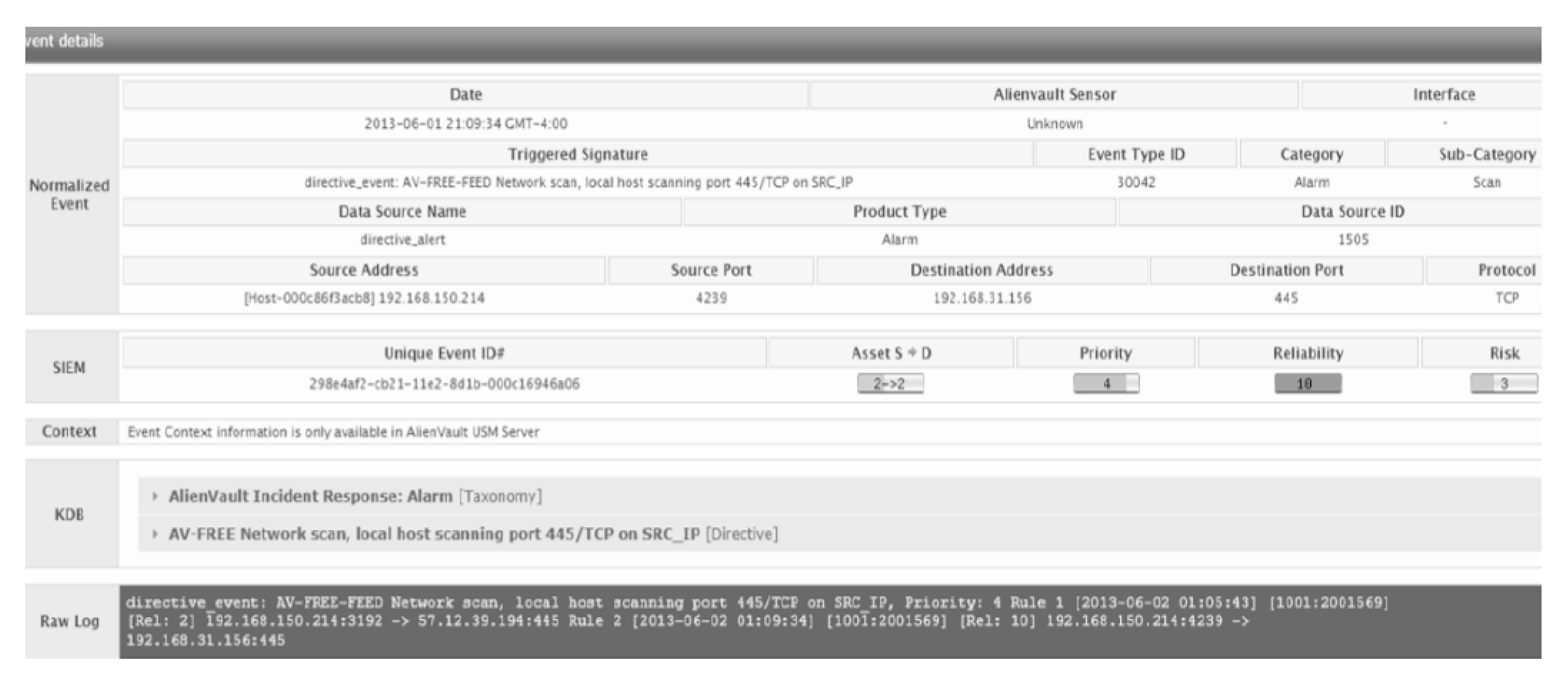


图 9-118　扫描 445 端口的日志

从图 9-119 可知，触发这类事件的特征是针对目标计算机的 445 端口的扫描，而且目的 IP 地址为随机分布，持续时间达到 48 分钟，所以风险等级这里为 3 级，图中用绿色方框表示出来，如果 Risk 等级上升，系统会用红色方框表示出来。

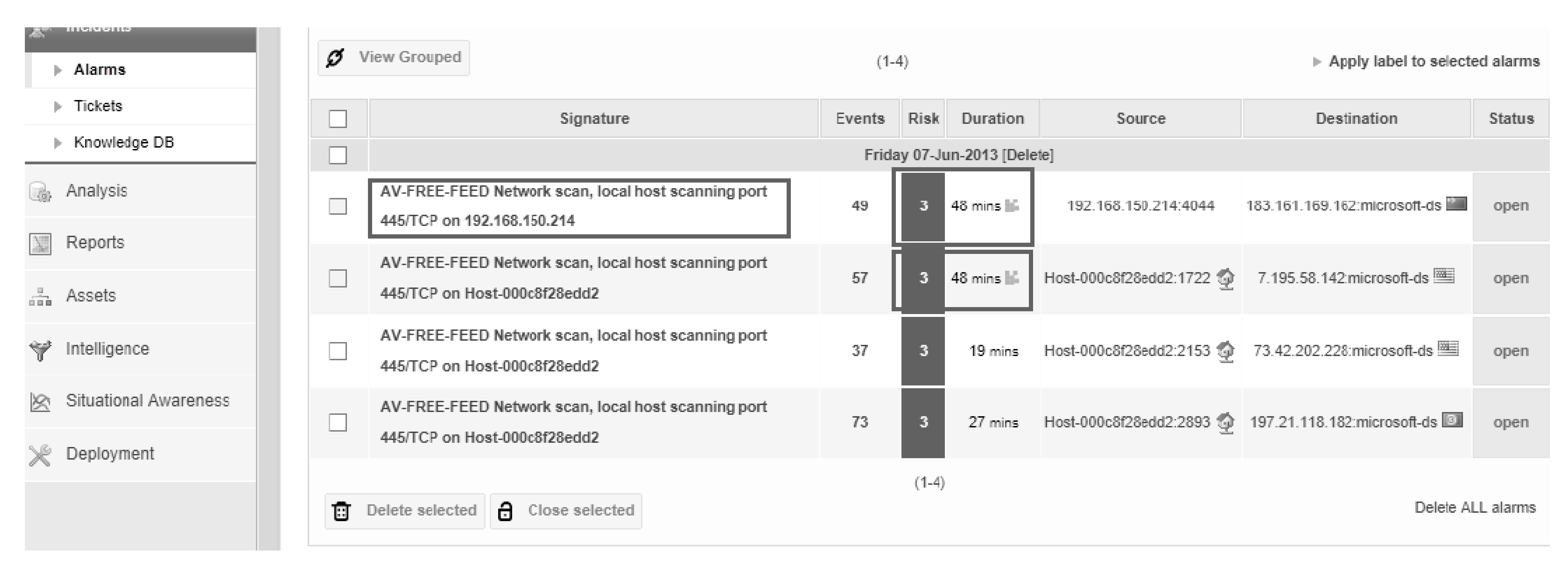
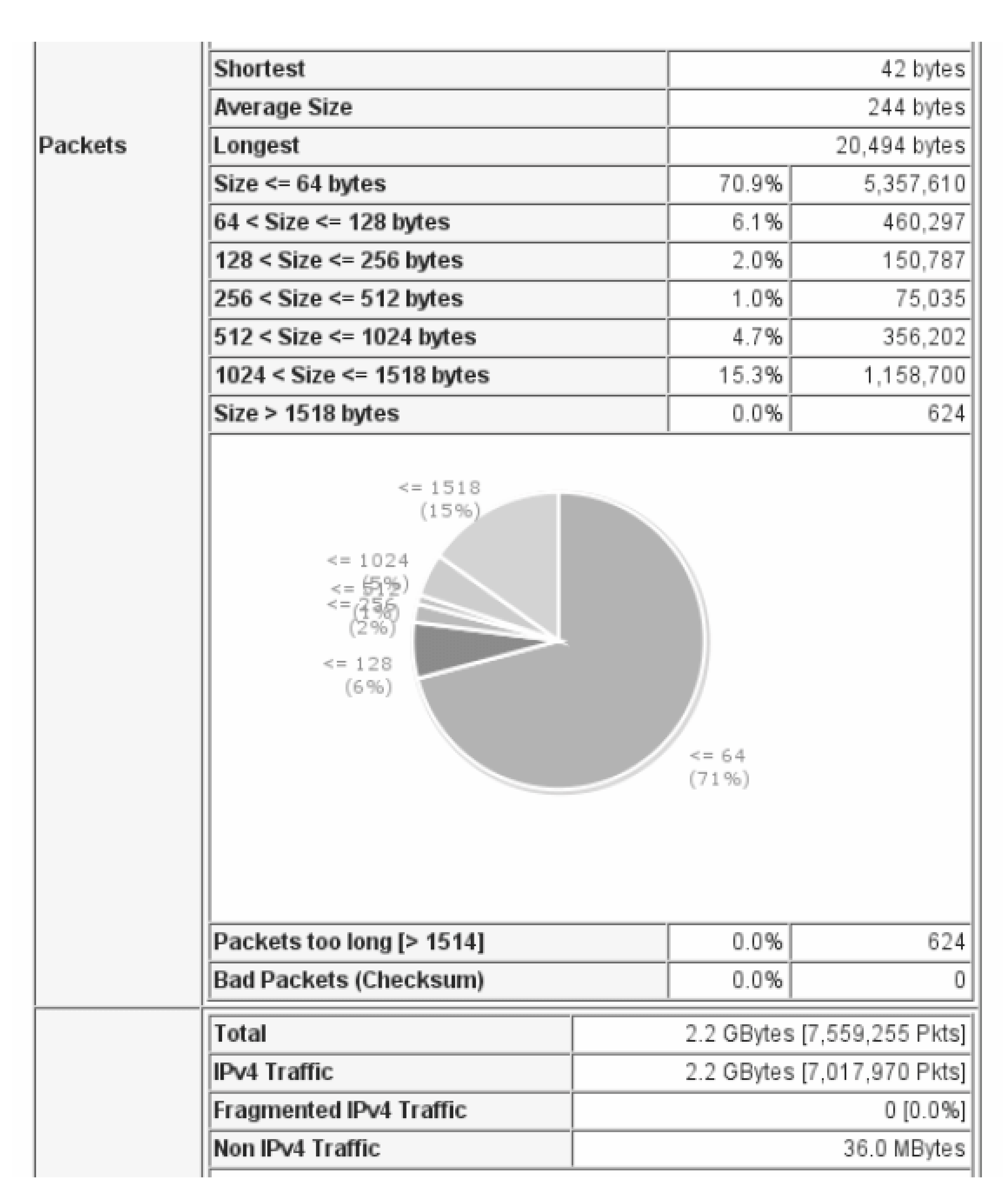


图 9-119　报告蠕虫扫描

9.10.4　流量分析

当蠕虫爆发时，在流量、协议以及数据包大小分布上都会与平时不同，此时可以借助于 OSSIM 系统中的 Ntop 流量监控软件来协助分析问题。Ntop 的部分截图如图 9-120 所示。



Packets	Shortest		42 bytes
	Average Size		244 bytes
	Longest		20,494 bytes
	Size <= 64 bytes	70.9%	5,357,610
	64 < Size <= 128 bytes	6.1%	460,297
	128 < Size <= 256 bytes	2.0%	150,787
	256 < Size <= 512 bytes	1.0%	75,035
	512 < Size <= 1024 bytes	4.7%	356,202
	1024 < Size <= 1518 bytes	15.3%	1,158,700
	Size > 1518 bytes	0.0%	624
	Packets too long [> 1514]	0.0%	624
	Bad Packets (Checksum)	0.0%	0
	Total		2.2 GBytes [7,559,255 Pkts]
	IPv4 Traffic		2.2 GBytes [7,017,970 Pkts]
	Fragmented IPv4 Traffic		0 [0.0%]
	Non IPv4 Traffic		36.0 MBytes

图 9-120　Ntop 检测到数据包大小分布

如图 9-121 所示，从图中的统计信息可以查看到，流量分布基本正常；数据包中 64~127 字节的数据包数约为 1024~1518 字节数据包数量的 3 倍，这说明当这种蠕虫爆发时，网络中小包数量过多，而且 SYN 连接数量非常大。从主机信息上观察，会发现很多陌生 IP，这些都是虚假 IP。

Host Information

Traffic Unit: Bytes

Subnet: All

Host	Location	IP Address	MAC Address	Community	Other Name(s)	Inbound vs Outbound
192.168.150.114		192.168.150.114	00:0C:29:75:70:A6			
192.168.150.200		192.168.150.200	00:0C:29:AF:56:17			
192.168.150.115		192.168.150.115	00:0C:29:16:29:AD			
192.168.150.2		192.168.150.2	00:50:56:F7:85:16			
192.168.150.174		192.168.150.174	00:0C:29:73:BE:F3			
192.168.1.80		192.168.1.80	00:50:56:F7:85:16			
192.168.1.87		192.168.1.87	00:50:56:F7:85:16			
192.168.1.89		192.168.1.89	00:50:56:F7:85:16			
192.168.1.92		192.168.1.92	00:50:56:F7:85:16			
192.168.1.74		192.168.1.74	00:50:56:F7:85:16			
192.168.129.119		192.168.129.119	00:50:56:F7:85:16			
192.168.1.127		192.168.1.127	00:50:56:F7:85:16			
192.168.129.98		192.168.129.98	00:50:56:F7:85:16			
192.168.129.109		192.168.129.109	00:50:56:F7:85:16			
192.168.1.20		192.168.1.20	00:50:56:F7:85:16			
192.168.129.23		192.168.129.23	00:50:56:F7:85:16			
192.168.129.31		192.168.129.31	00:50:56:F7:85:16			
192.168.129.3		192.168.129.3	00:50:56:F7:85:16			

图 9-121　无流量的虚假 IP 信息

接下来，谈谈 IP 碎片攻击对 IDS 的危害。当路由器准备将 IP 分组发送到网络，而该网络又无法将这个 IP 分组一次全部发送完时，路由器必须将该分组分成小块，使其长度能够满

足这一网络对分组大小的限制，这些分割出来的小块就叫碎片（fragment）。因此，小 IP 报文又称为 IP 碎片。IP 碎片可以独立地通过不同的路径转发，而且碎片不一定按照次序到达。当到达目的主机后，目的主机会重组 IP 碎片。不过，现在的 IDS 产品基本都具有良好的 IP 碎片重组能力，因此基本的 IP 碎片问题不会给 IDS 造成太大的麻烦。但是，像碎片重叠之类的技术仍会带来很大的问题。

有的攻击者，就利用了碎片重叠方法实施攻击。当两个碎片有部分数据重叠时，其中一个碎片的数据会覆盖掉另一个碎片的重叠数据。但是，至于哪一个碎片重叠部分的数据被覆盖，则由操作系统类型决定。例如，在 Windows 2000 和 Solaris 2.x 系统中，若碎片 frag1 先于 frag2 到达，frag1 的数据会覆盖掉 frag2 的数据重叠部分。若两个碎片不按正常顺序到达，即 frag2 先于 frag1 到达，frag2 的数据会覆盖 frag1 的数据重叠部分。在本书 5.2.4 小节介绍过利用 Fragroute 测试小包攻击。

9.10.5　协议分析

根据上文介绍的方法，我们基本定位了网络故障来源和分类，下面就可以根据分析数据包来查看这种蠕虫，我们通过集成在 OSSIM 系统中的基于 Web 的抓包工具来有针对地对数据包进行解码（Decode），如图 9-122 所示。

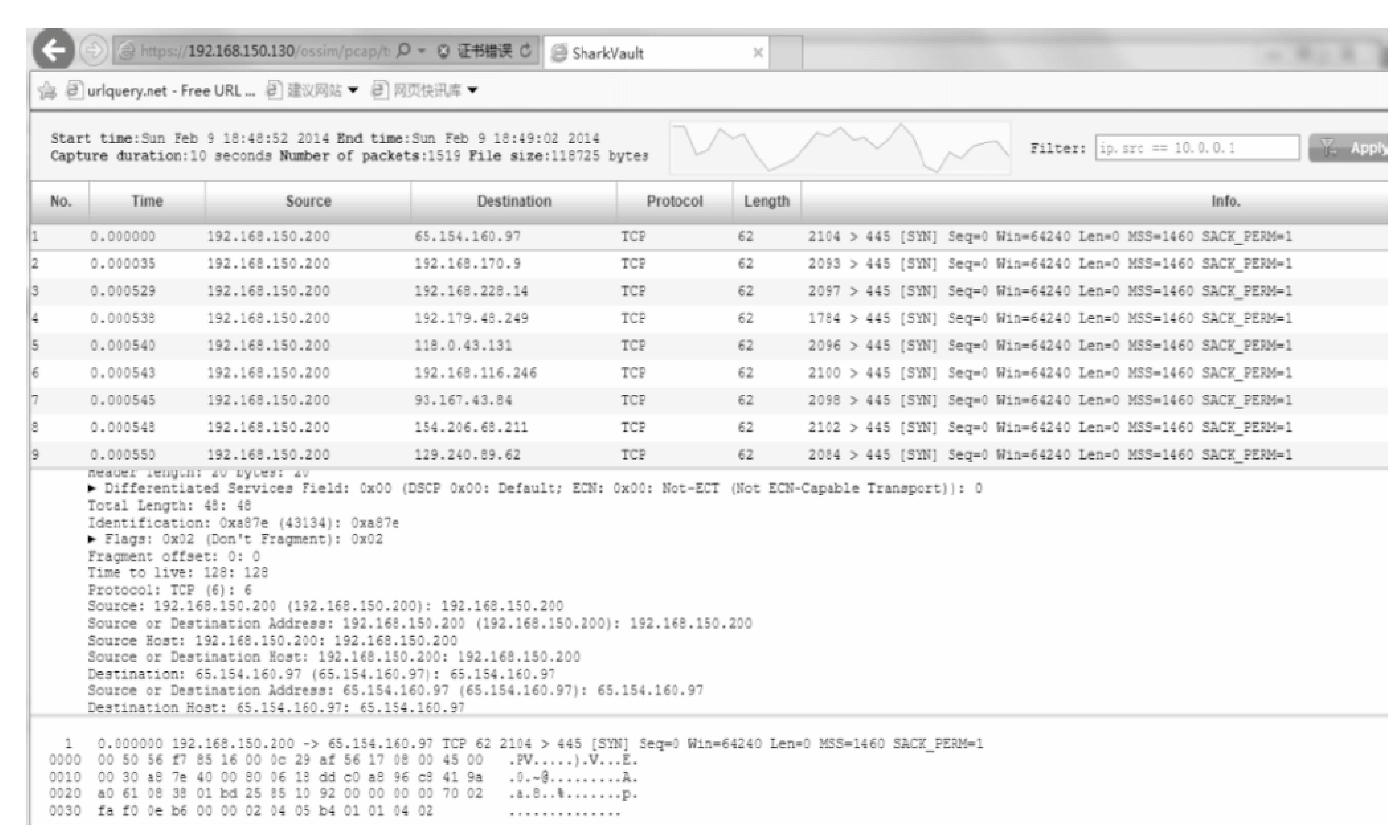


图 9-122　利用 OSSIM 进行协议分析

在主体窗口中显示了抓包的情况，列出了抓到数据包的序号、时间、源目的 MAC 地址、源目的 IP 地址、协议类型、源目的端口号等内容。很容易看出 IP 地址为 192.168.150.200 的主机，在极短的时间内向大量的不同主机发出了访问请求，并且目的端口都是 445。

图 9-122 可以看出，这些数据包的长度都是 62 个字节。数据包前 12 个字节包括了目的 MAC 和源 MAC 的地址信息，紧跟着的 2 字节指出了数据包的类型，0800 代表的是 IP 包格式，0806 代表 ARP 包格式。接着 20 个字节是封装的 IP 包头，包括了源、目的 IP 地址、IP 版本号等信息。剩下的 28 个字节封装的是 TCP 包头，包括了源、目的端口，TCP 链接的状态信息等，这就构成了一个 62 字节的包。可以看出除了这些包头数据之外，没有携带有效数据负荷，所以这是一个 TCP 要求 445 端口同步的空包，也就是病毒主机在扫描 445 端口。一旦染毒主机同步上没有采取防护措施的主机 445 端口，便会利用系统漏洞传播病毒。

正常情况下访问主机有可能在这么短的时间里发起这么多的访问请求吗？通常端系统一般都会在发送的数据段获得对端的确认之后，才会主动发送 FIN 报文，释放 TCP 连接，如果在多次重传之后仍未得到对端的确认，通常会向对端发送 RST 报文，异常释放 TCP 连接，所以在毫秒级的时间内发出几十或更多连接请求就属于不正常。

9.11 时间线分析方法

通常情况下，当攻击者入侵时，他们并不是瞄准某个信息系统，他们通常渗透后先建立一个支点，然后逐渐深入网络的某主机，直到找到所需要的信息。尤其在网络蠕虫爆发时，其攻击间隔一般小于 1 秒，如何分析发展过程和趋势呢？一般通过抓包软件很难掌握网络蠕虫整体爆发情况，OSSIM 中提供的 SIEM 控制台提供了时间线分析工具，该功能有点类似于 Splunk 这款工具中的时间线功能，它反映出了攻击频率，是一种比较直观的分析工具。

9.11.1 时间线分析法的优势

SIEM 控制台是 OSSIM 查看事件的平台，它可以规范化地展示从远端采集的数据，在这里可以实现各种数据的过滤显示，提供的时间线分析工具对分析故障尤为重要，其特点如下：

- 时间线将离散的数据通过时间关联曲线进行聚合。
- 图形化的时间线显示，更容易找出未知事件之间的关联，以帮助挖掘出一些奇怪的问题。
- 利用不同颜色更容易做到感兴趣日志的突出显示。

9.11.2 实例

有明确意图的攻击总是由一系列事件顺序组成的，下面我们对照图片，来讲解如何使用 Timeline。对于拒绝服务类攻击，日志显示间隔可能是秒级的，这些日志仅靠人工查看费力不讨好。我们可以利用时间线分析工具（Timeline Analysis Tools）来解决这个问题。使用方法是选择 Analysis→SecurityEvents（SIEM），单击 Timeline Analysis 按钮，效果如图 9-123 所示。

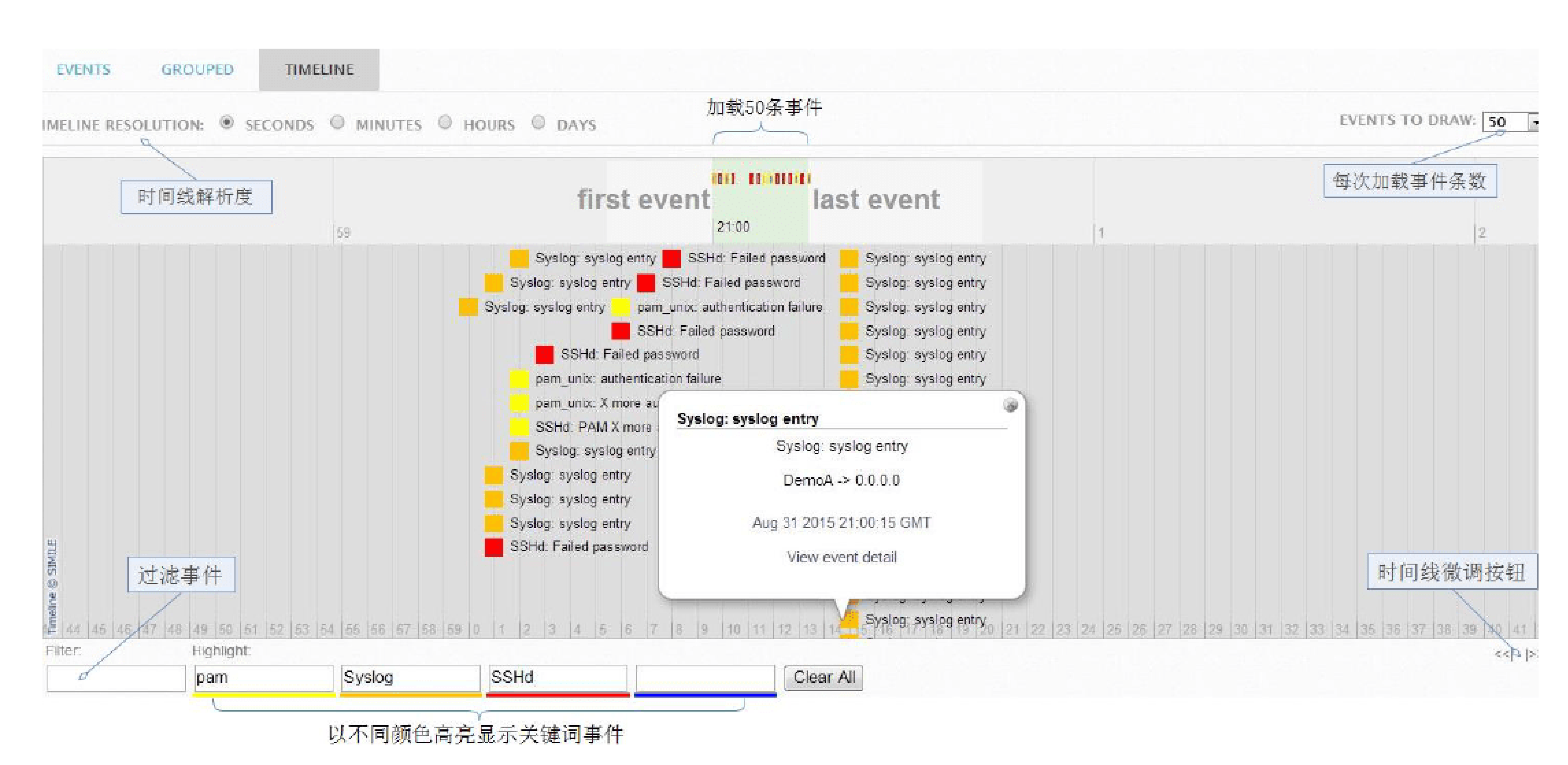


图 9-123　时间线分析

实践中还尝试诸如 Win32.Lioten.KX、狙击波（Worm.Zotob.A）、Backdoor/SdBot.ce 等攻击 139、445 端口的蠕虫。由于蠕虫具有自动化、迅速传播机制，传播速度非常迅速，而且扫描和攻击之间的时间间隔非常小，根本无法进行人工干预，这种情况下利用关联规则就能将这种攻击进行告警，在局域网中某台机器如果在一段时间内连接了几百台主机的 445 端口，这种情况就很有可能是蠕虫扫描攻击。再看一个例子，如图 9-124（间隔为秒）所示。

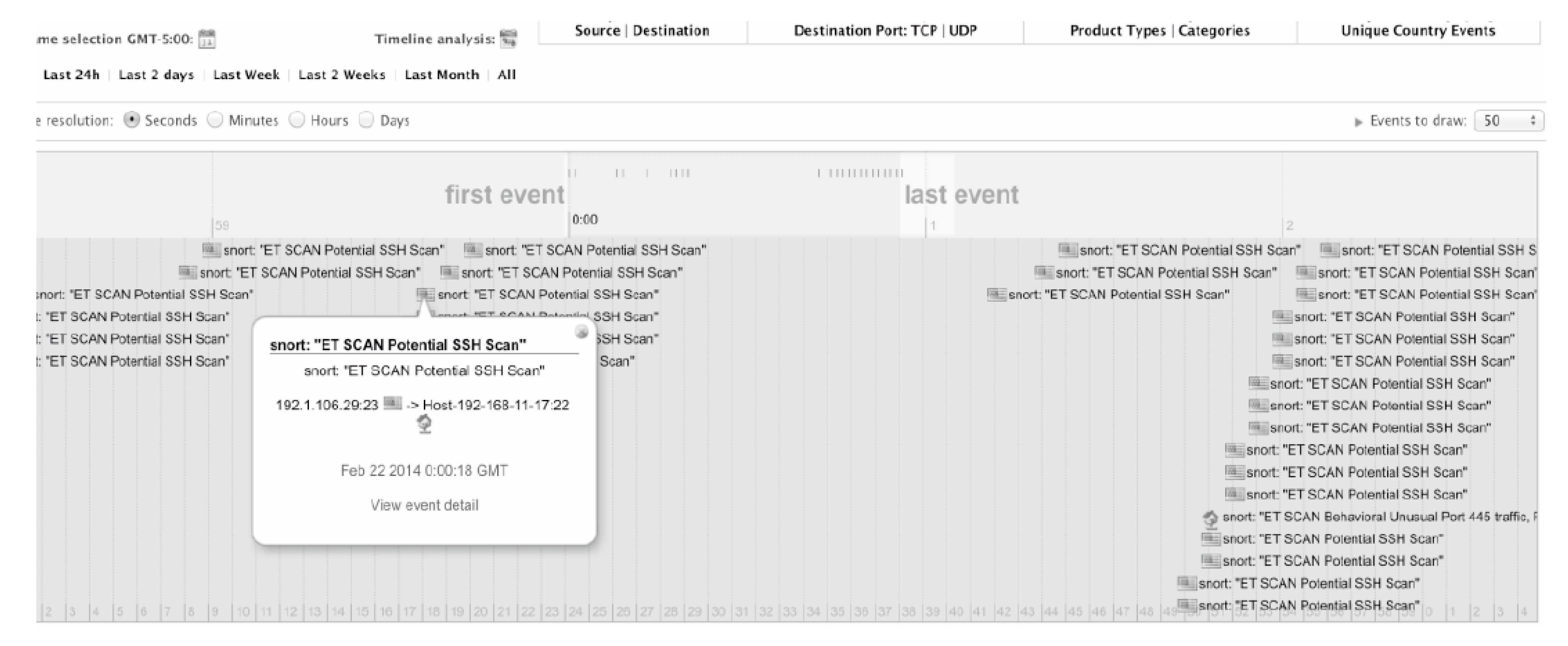


图 9-124　时间线分析 SSH 扫描

常规方法中，如果说蠕虫难于分析和捕捉，那下一节将要介绍的 Shellcode 攻击就更加难以监测。

9.12 利用 OSSIM 进行高级攻击检测

如今 APT 已经对信息安全带来了巨大挑战，目前所常用的 Nagios、Ntop、zabbix、Cacti 等单独使用某个流量监控工具很难察觉 APT 攻击，所以采用具有 SIEM 功能的 OSSIM 平台就是对付 APT 最有效的工具，因为这个系统可以从不同来源收集和关联安全数据，它们可以帮助你从海量信息中找出 APT 攻击渗透入网络的踪迹。

APT 攻击通常是一种持续的攻击活动，而非单一的攻击事件。在攻击过程中，它会利用各种手段不停地尝试，直到达到目的。许多攻击开始阶段，伪装成合法流量进入内网，得逞之后再尝试进行破坏。如果你仅仅用 Cacti、Zabbix 这类工具，而不进行数据包深度检测，很难发现这种隐匿的攻击。

在 APT 攻击中，攻击者经常会对目标网络及应用针对性地进行为期几个月甚至更长时间的渗透，他们会想尽办法利用网络中应用程序漏洞形成攻击者所需 C&C 网络，当一切就绪，攻击者会在某台服务器中部署 RootKit，便能通过精心构筑的 C&C 网络回传目标文件。那么防范 APT 攻击，最好的方法是数据包检测，尤其是内网数据包的深度检测，因为那些攻击数据包会进入网络内部开展破坏和攻击行为，对 APT 而言，传出流量更具危险性，所以采用 OSSIM 对传出/传入流量进行异常检测就是一种行之有效的方法。

在网络攻击中还有不少是属于 Shellcode 攻击，但往往很多管理员对其不了解，在网络攻击过程中，基于特征的 IDS 系统往往也会对常见的 Shellcode 进行拦截。但一些高级的 Shellcode 经过乔装打扮蒙混过关。这个过程就好比一枚炮弹飞向目标的过程。炮弹的设计者关注的是怎样计算飞行路线，锁定目标，最终把弹头精确地运载到目的地并引爆，而并不关心所承载的弹头里到底装的是沙子还是核弹。Shellcode 在网络监测中容易忽视，但其危害巨大，其实 Shellcode 是在渗透时作为载荷运行的一组机器指令，它通常用汇编语言编写，在如果监控网段存在 Shellcode 攻击则会被记录，在 OSSIM 的 SIEM 控制面板中可以明显看到此次攻击。

9.12.1 误用检测与异常检测

OSSIM 中的事件分析器使用了误用检测（Misuse detection）与异常检测（Anomaly Detection）两种技术。下面我们分别介绍这两种检测技术的特点。

1. 误用检测（Misuse detection）

误用检测是基于特征码检测技术（Signature-based detection），系统的目标是检测主体活动是否符合这些模式。所以最适合已知模式的检测，系统里使用了简单匹配（Pattern Matching）以及专家系统。例如 Snort 特征码，“ET POLICY PE EXE or DLL Windows file download”达到一定数量需要引起我们注意，下面我们观察在 SIEM 中的事件，如图 9-125~图 9-127 所示。

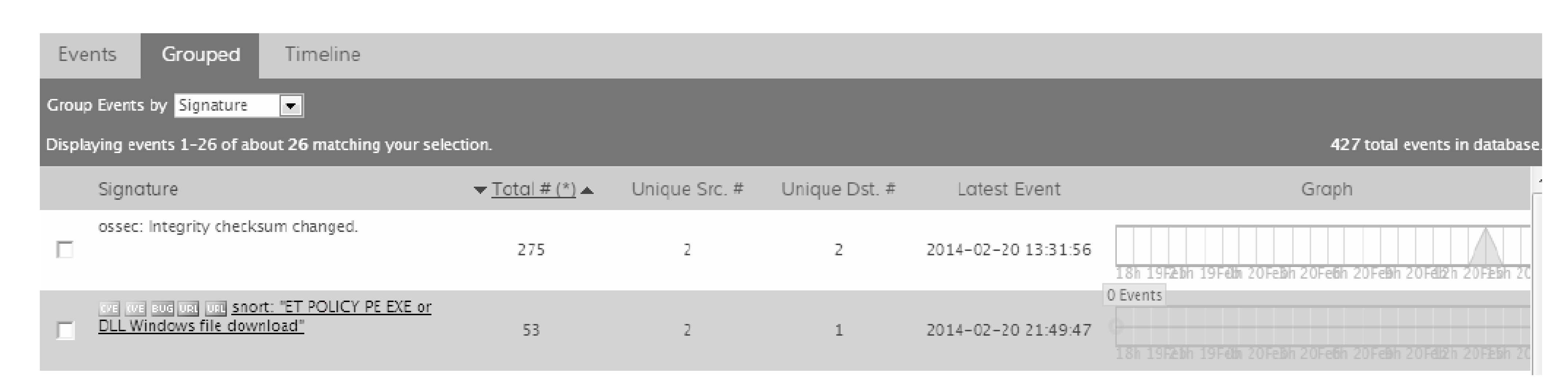


图 9-125　在 SIEM 中发现可疑文件下载

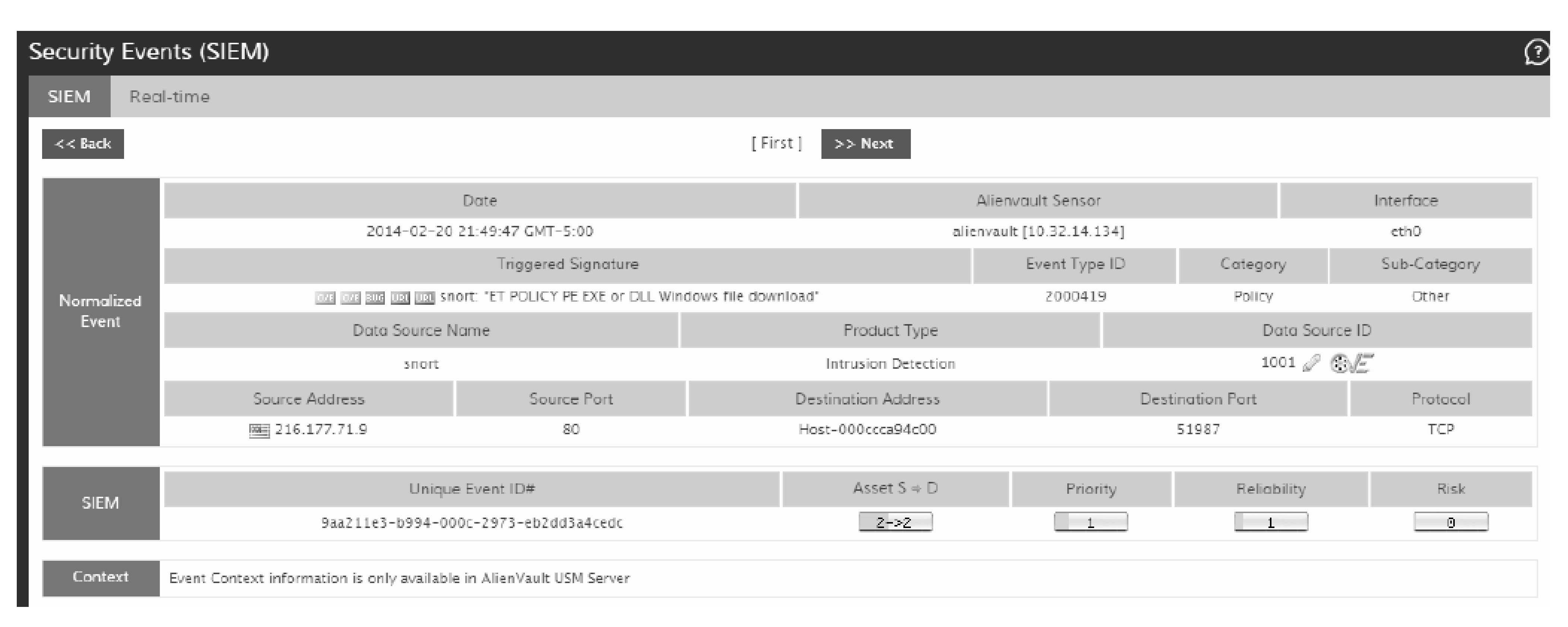


图 9-126　分析报警内容

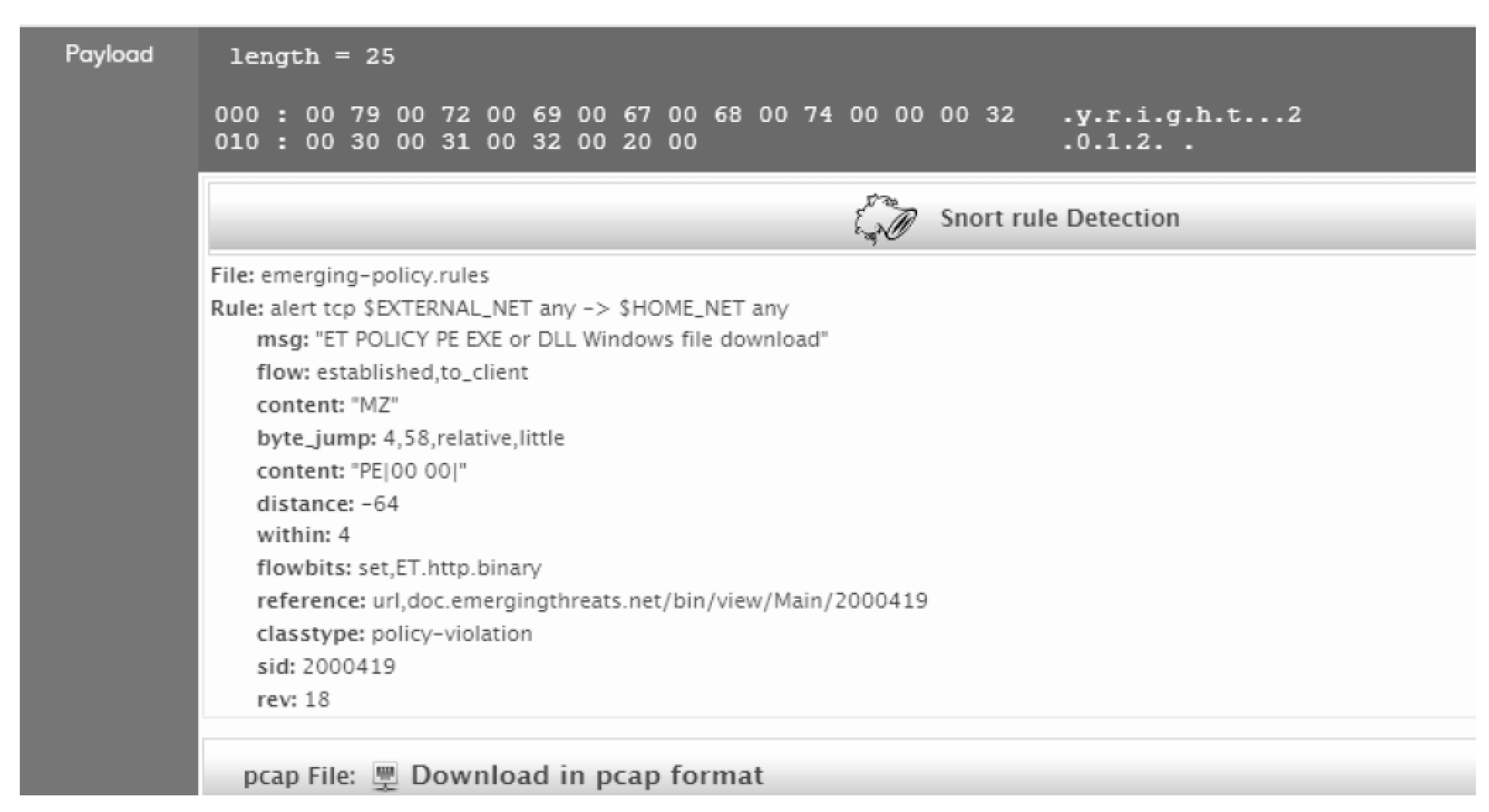


图 9-127　分析有效载荷

2. 异常检测（Anomaly detection）

本书第 6 章介绍的 snort 是基于特征检测的 NIDS，它能根据事件特征（signature）准确识别已知攻击行为，但攻击者会将恶意流量进行分片、压缩、编码以规避 NIDS 检测，此时遇到了一定技术瓶颈，要更好地检测出未知攻击，还要使用异常检测手段。

异常检测技术通常先构建正常用户行为集合，再比较被监测系统实际行为模式和正常模式之间的偏离程度，最终确定是否属于入侵。OSSIM 中的异常检测技术用到“模式比较”和“聚类”等算法，通过关联规则建立正常行为模式，然后通过行为比较判定攻击。这样，不需要过多依赖系统的相关知识，具有较强的适应性。

优点是可以检测新的入侵行为，缺点是误报较多。大多数的正常行为的模型使用一种矩阵的数学模型，矩阵的数量来自于系统的各种统计指标，比如 CPU 使用率、内存使用率、登录时间、次数、IP、网络活动及文件改动等，在 OSSIM SIEM 控制台中通过 DATA SOURCES 列表中选择“anomalies”能观察到实例如图 9-128、图 9-129 所示。

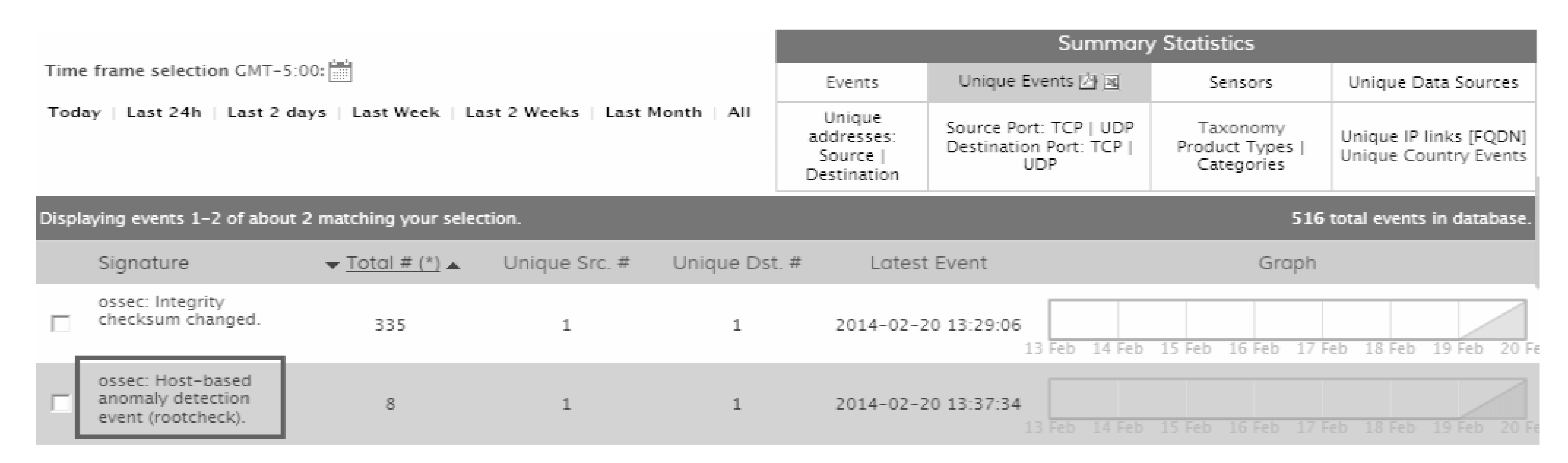


图 9-128　由 OSSEC 检测到异常

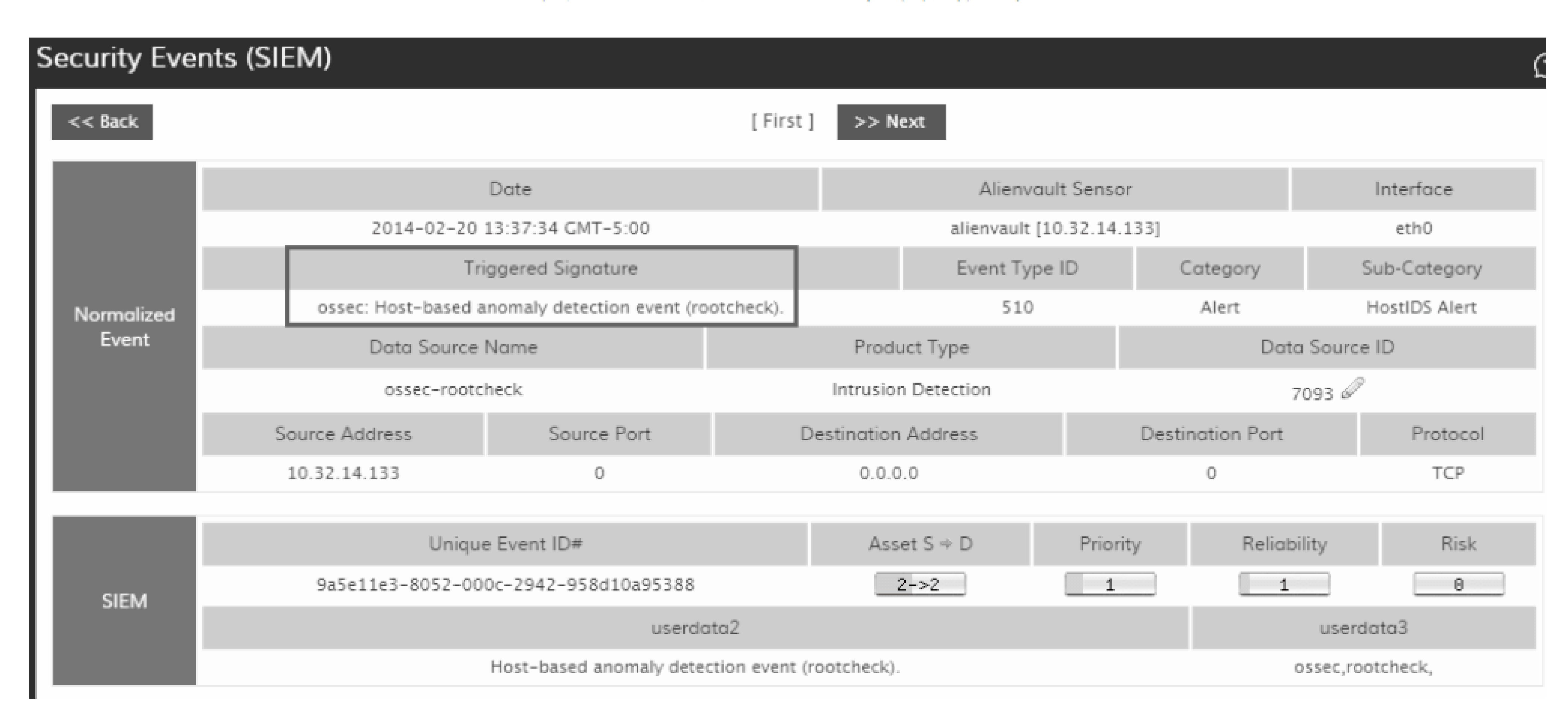


图 9-129　查看异常报警内容

接着我们在 SIEM 控制台中利用时间线分析工具查看 Shellcode 的具体实例，如图 9-130 所示，我们可以选中某个 Shellcode 日志查看详情。

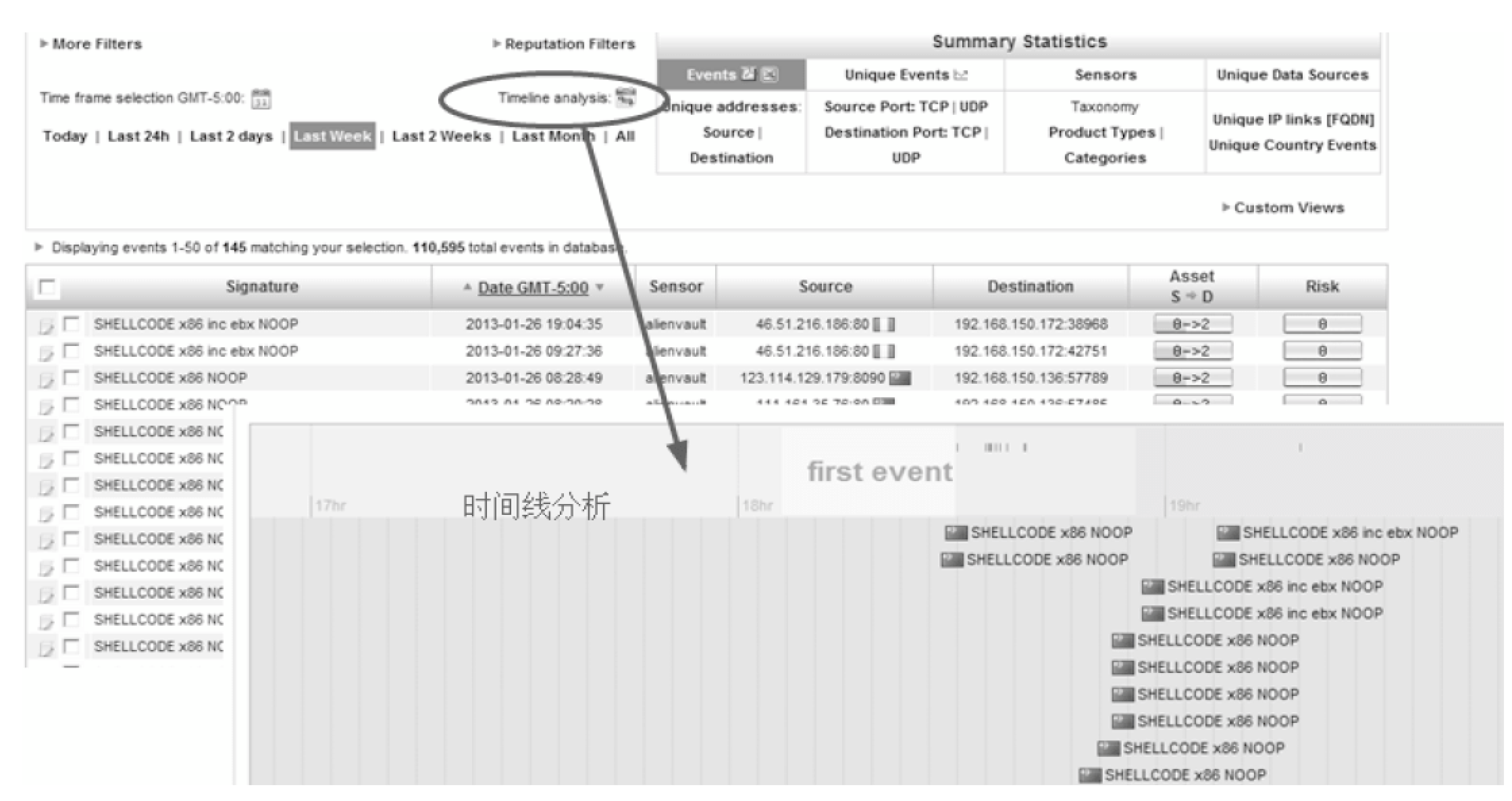


图 9-130　用时间线分析攻击

收到这个消息时的 Shellcode 事件有可能是误报。大多数这些规则是通用的，可能会得到大量的误报。

问题来了，既然有异常检测还有特征检测，还是会伴随误报的情况，为何？因为网络入侵检测本身就具有不确定性，这就使得误报或漏报成为必然。无论是异常检测还是特征检测，都必有误报或漏报，异常检测以正常活动的特征轮廓为基准，但我们并不能保证不符合正常活动轮廓的网络行为就是真实攻击，就像上面的例子中也有些 Shellcode 攻击属于误报（False Positive）。

反过来理解，我们也不能保证符合正常活动的行为就一定不是攻击。特征检测的基础是恶意流量符合独特的模式，而正常或良性流量不符合，但我们并不能保证所有的正常流量都不符合，即不能保证特征与恶意流量的一一对应关系，也有可能有些无恶意的正常的数据包也符合这一特征。

9.12.2　绘制 Shellcode 代码执行流程图

程序执行流程我们会在一些反编译工具中看到，在 OSSIM3.x 版本中也有这样的功能，在图 9-131 中绘制的图像能显示以下信息：

- 装载运行指令的过程，一共多少指令;
- 在这个界面中绘制出刚刚执行过的指令块及其调用关系;
- 将刚读入的指令块读入后，观看静态代码的同时了解程序动态执行的流程。

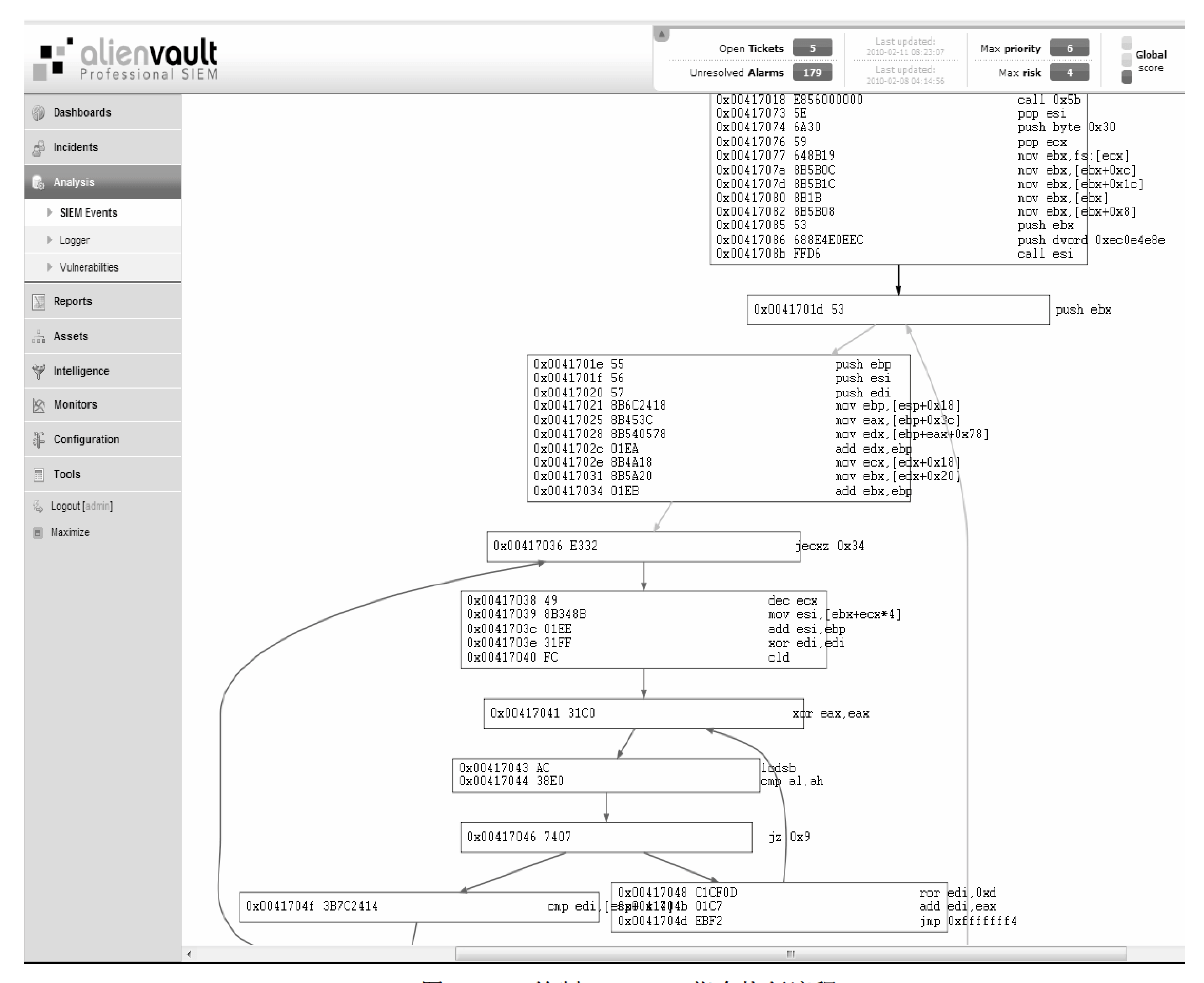


图 9-131　绘制 Shellcode 指令执行流程

9.12.3　收集异常行为流量样本

在分析网络异常流量和服务器异常行为时，仅仅根据 SIEM 控制台中报告的各种安全告警事件还不够，安全人员还需要抓取异常流量的封包样本分析 Payload，在此之前，工程师们常常额外架设一套抓包系统，现在通过 OSSIM 的 Suricata 就可以实现保存 HTTP 流量中的指定后缀或者特定格式的文件，比如 jpg、pdf、exe 等。图 9-132 所示为异常事件中保存的 Payload，随时可以下载，这个特性对希望抓取恶意软件样本的安全研究人员非常有用。

PCAP FILE: DOWNLOAD IN PCAP FORMAT

PAYLOAD

GENINFO
FRAME
IP
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 76
Identification: 0x2407 (9223)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Source: 192.168.11.105 (192.168.11.105)
Source or Destination Address: 192.168.11.105 (192.168.11.105)
Source Host: 192.168.11.105
Source or Destination Host: 192.168.11.105
Destination: 192.168.11.7 (192.168.11.7)
Source or Destination Address: 192.168.11.7 (192.168.11.7)
Destination Host: 192.168.11.7
Source or Destination Host: 192.168.11.7

图 9-132 下载 Payload

9.13 合规管理及统一报表输出

9.13.1 合规管理目标

通过对收集的各种设备日志和安全事件进行深入挖掘，结合安全合规管理，达到帮助建设企业信息安全合规管理体系的目标。OSSIM 系统能够生成不同的安全合规报表，通过检查信息系统中的各种安全合规项信息，生成安全合规测试报告，以达到对安全合规管理的要求，帮助企业规避风险。

9.13.2 主要技术

OSSIM 系统中可以根据用户的需要，将各种日志以及已检测到的流量和入侵行为，动态地生成各种类型的报表，通过 Web 的方式实现报表的预览与打印。该功能就是由通用报表模块实现完成，也就是常说的 BI（商业智能）系统中的报表模块，报表模块的核心是报表生成引擎 Jasper Reports，它采用了开源项目，是一个强大、灵活的报表生成工具，能够展示丰富的内容，并将之转换成 PDF、HTML、XLS、CSV 或者 XML 格式。

在 OSSIM 有些商业版本中利用 Jasper Reports+JFreeChart 的模式做统计报表分析。它完全采用 Java 编写，在 OSSIM 2.x 和 3.x 版本中这些报表信息存放在 jasperserver 数据库中，到了 OSSIM 4.3 版本采用了新的架构，采用 Jgraph 技术，如果将 Tomcat 服务关闭报表就会停止输出。

在 OSSIM 系统所有报表（包括饼状图、柱状图、曲线图、雷达图等）的升级都是采用 Jasper Reports 设计的强大而直观的可视化报表设计器，而开发 Jasper Reports 的工具叫做 iReport

（http://community.jaspersoft.com/）。这个报表工具允许用户可视化编辑包含 charts 等复杂的报表。它还集成了 JFreeChart 图表制作包，允许用户可视化地编辑 XML JasperDesign 文件。用于打印的数据可以通过多种方式获取，包括 JDBC、TableModels、 JavaBeans、XML、Hibernate（支持 HQL 查询语言）、CSV 等。它支持多种输出格式，包括 PDF、RTF、XML、XLS、CSV 及 HTML。

在日志报表输出方面，OSSIM 系统中可以输出 Alarm、Asset、Availability、Business & Compliance ISO PCI、Geographic、 Metric、SIEM Events、Tickets、Vulnerabilities、User Activity 十几个大类，其中每个大类还可以细分若干子类。

9.13.3 什么是合规

安全管理中要求所有日志的归档都需要遵循法规遵从（Compliance）原则。Alienvault USM 可帮助企业满足 PCI-DSS（支付卡行业数据安全标准）中 R1~R12 各项条款的要求。例如，第 10 款中要求支付服务提供商必须记录系统日志，以便追踪并报告对其网络资源和持卡人数据的所有访问。网络环境中的日志，可在出现问题的时候用于刑侦分析。如果没有系统活动日志，那将很难找到故障原因。

在 OSSIM 中查看方法为 Threat Intelligence→Compliance Mappging，如图 9-133 所示。

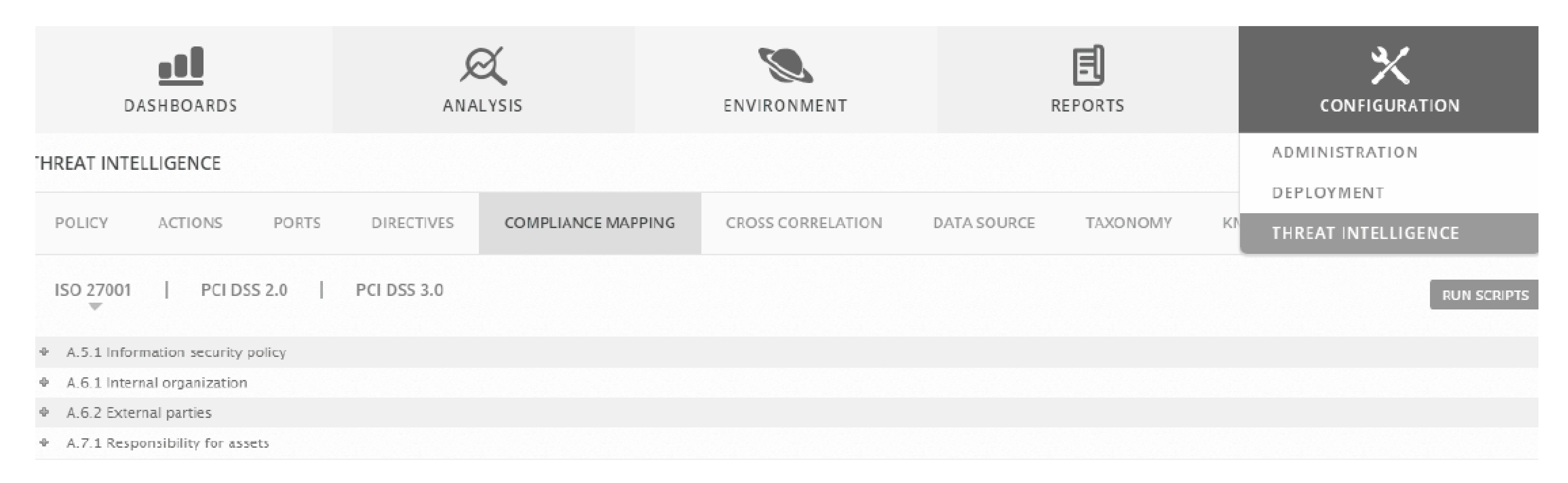


图 9-133 合规性查询

合规运行脚本位于/usr/share/ossim/compliance/scripts/datawarehouse/目录之下，程序文件主要为 Perl 脚本。

9.13.4 理解 PCI 合规遵从

OSSIM 中的合规遵从数据源来源于 SIEM 控制台的事件以及 Logger，我们在 OSSIM USM 平台的 Configuration→Threat Intelligence→PCI DSS 2.0 中可以查看到，如图 9-134 所示。

图 9-134　PCI 报表展示内容

PCI 合规遵从分为六大类，共 12 个需求，详情如下：

（1）第一类：建设和维护安全网络

- Requirement 1 – Install and maintain a firewall configuration to protect cardholder data，安装和维护防火墙配置，以保护持卡人数据，说明组织必须有批准和测试所有外部网络连接以及防火墙配置变更的正式过程。我们必须验证防火墙配置变更得到了授权，避免非法变更的情况。那么就需使用防火墙日志来记录配置变更。
- Requirement 2 – Do not use vendor–supplied defaults for system passwords and other security parameters ，这里讨论密码管理的实践和安全加固。比如不启动没有必要的服务，日志条目可以告诉我们，在何时启动了事先关闭的服务。

（2）第二类：保护持卡人数据

- Requirement 3 – Protect stored cardholder data，需求 3 要更进一步处理数据加密，例如在 R.3.6 小节，要求验证这些活动是否真正进行的日志，在大部分加密系统记录密钥生成分发的日志，需要记录。
- Requirement 4 – Encrypt transmission of cardholder data across open, public networks，同样还是处理加密。

（3）第三类：确保脆弱性管理体系的维护

- Requirement 5 – Use and regularly update anti–virus software，这是针对病毒的防御，在 R.5.2 小节中记录了要确保所有防病毒工具是最新且处于活动状态，而且要能生成审计日志。如果公司的防病毒系统更新失效，意味着其网络可能遭受恶意软件的攻击。
- Requirement 6 – Develop and maintain secure systems and applications，这里要求组织开发和维护安全的系统和应用程序。

（4）第四类：实施强访问控制

- Requirement 7 – Restrict access to cardholder data by business need–to–know，这里根据业务按需了解的原则限制对持卡人数据的访问，需要日志来验证谁能够访问所指的数据。
- Requirement 8 – Assign a unique ID to each person with computer access，这一需求最详细，就是为每个具有计算机访问权的人分配唯一的 ID，而且具有一定权限。人们必须控制用户 ID，在 R8.5.9 中规定了必须在 90 天内至少修改一次用户密码，也可以通过审核来自服务器的日志来验证。
- Requirement 9 - Restrict physical access to cardholder data，限制物理访问持卡人数据，将对系统组件和持卡人数据的访问限制为只有工作需要访问这些数据的人。为每位拥有计算机访问权限的用户分配唯一的 ID。限制对持卡人数据的物理访问。在 R.9.4 中规定，使用访问者日志维护访问者活动，并保留这些日志至少 3 个月，除非法律有其他限制。

（5）第五类：经常性地监控和测试网络

- Requirement 10 - Track and monitor all access to network resources and cardholder data 跟踪并监控所有对网络资源的访问和持卡人的数据。
- Requirement 11 - Regularly test security systems and processes 定期测试安全系统和过程，说明扫描范围内系统漏洞测试的必要性，在 R.11.4 中要求使用 IDS/IPS：使用网络入侵检测系统和入侵防御系统监控所有网络流量，并向有关人员发出可疑入侵的报告，确保入侵检测引擎为最新。

（6）第六类：确保信息安全策略的维护

- Requirement 12 – Maintain a policy that addresses information security，这是最高要求，它涵盖了安全策略、标准和日常运营的规程。尽量按照这个要求来对网络系统进行管理，通过精心收集、分析数据，才可能满足 PCI DSS 的众多要求。接下来，我们还需要了解 OSSIM 的多种报表合规格式。在合规展示方面，除了通过报表的方式，还可以在仪表盘中对这十二项要求实时更新数据，如图 9-135 所示。

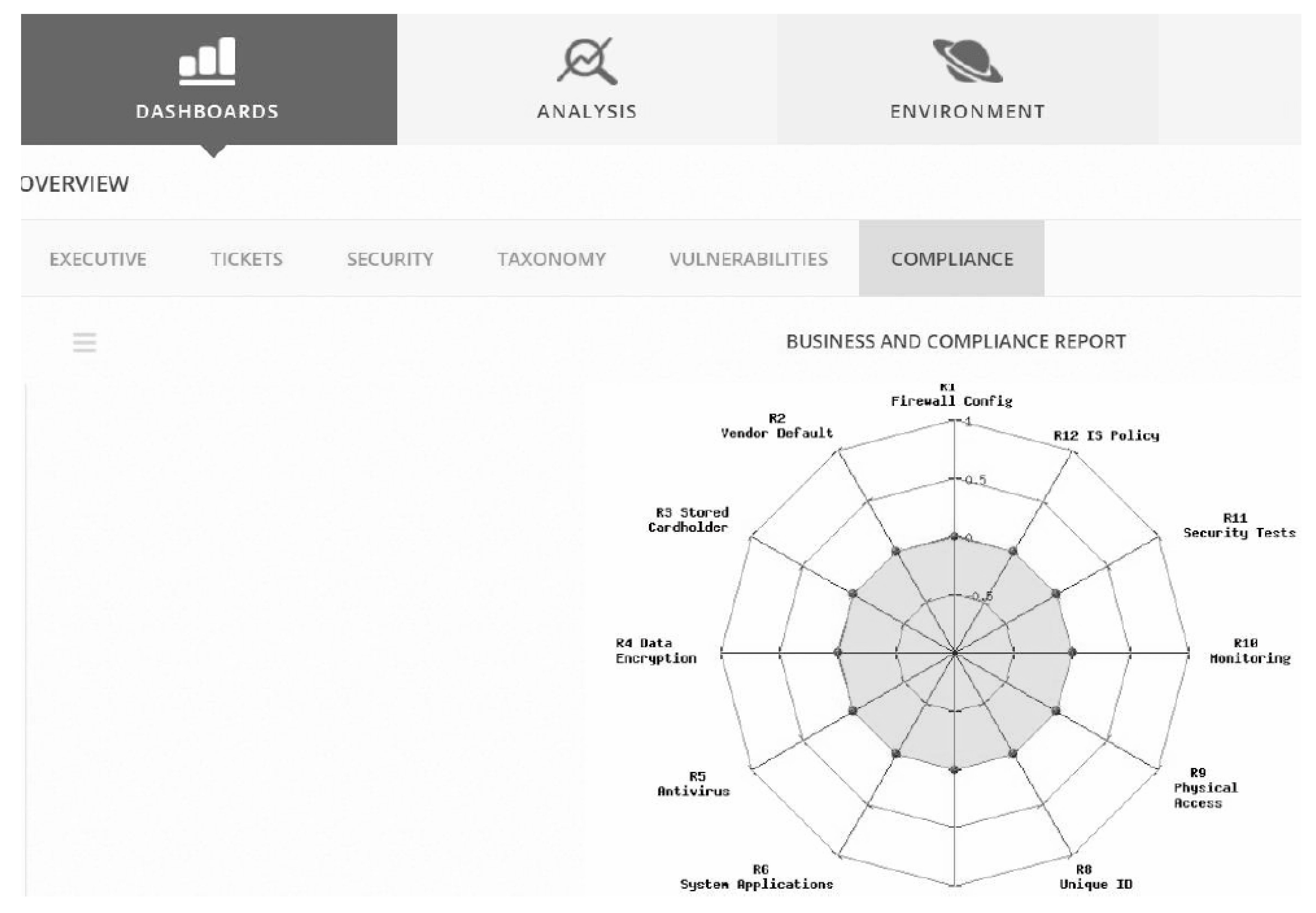


图 9-135　仪表盘显示的合规检测结果

9.13.5　报表类型

1. OSSIM USM 报表

在开源版 OSSIM 中输出报表有限，主要以 PDF 形式输出，而 OSSIM 企业版提供比开源版更丰富的报表，包含更多细节，能在线预览，也能以各种文件格式下载。表 9-6 总结了 OSSIM USM 中输出报表种类。

表 9-6　USM 报表

编号	报表名称	细　节	时　间
1	Alarm 报告	Top Attacked Host Top Destination Ports Top Alarm Top Alarms by Risk	30 天
2	资产报告	资产摘要、报警 漏洞、安全事件、裸日志等	30 天
3	可用性报告	可用性趋势报告、可用性状态、事件、性能等报表	30 天
4	商业和法规遵从	风险、PCI DSS 2.0、3.0 报表	30 天
5	数据库活动	数据库安全事件、日志报表	30 天
6	数据源事件	由数据源分类的事件报表	30 天
7	数据产生类型	根据事件产生类型分类的统计表	30 天

（续表）

编号	报表名称	细 节	时 间
8	FISMA 报表	用户认证，活动访问等安全事件，攻击日志分类的日志报告、PCI 无线报告以及 Tickets 状态	30 天
9	地理位置报表	按 IP 出现的国家分类报告	30 天
10	HIPAA 报表	Alarm 攻击分类的 Top 10，裸日事件信息，安全事件 Top10 等报表	30 天
11	Honeypot 活动报表	数据源分类的事件，各类安全事件，根据数据源分类的不同特征码	30 天
12	ISO 27001 技术报告	ISO27001-A.10.4.1 ISO 27001-A.10.6.1 ISO27001-A 10.10.1	30 天
13	Malware 告警	Alarm Top 攻击主机排名及列表	30 天
14	PCI 2.0 报告	全面（包括主机、防火墙、认证、无线、加密传输等方面）测评报告	30 天
15	PCI3.0 报告	比 PCI 2.0 更详细的报告	30 天
16	策略配置和变更报告	系统安全配置、认证配置策略变更报告	30 天
17	日志访问报告	主机攻击事件、目标端口访问事件、应用访问事件、IPS/IDS 事件、邮件服务器事件、路由器服务器、VPN 事件	30 天
18	Raw Log	Alarm、Alert、异常行为检测、病毒、应用、防火墙、认证、DHCP、可用性、数据保护、蜜罐、IDS/IPS、Inventory、邮件安全、邮件服务器、Malware、管理平台、网络发现、Web 服务器、渗透、无线及漏洞扫描等日志报告	30 天
19	安全事件	安全访问事件、账户改变、Alarm、Alert 等同 Raw Log 的报表分类	30 天
20	SOX 报告	Tickets、Alarms 及安全事件报告	30 天
21	用户活动	用户活动报告	30 天
22	漏洞报告	漏洞报告（严重、高、中、低）	30 天

2. OSSIM 报表

开源的 OSSIM 报表种类和数量相对简单，可提供 Alarms 报告、资产报告、可用性报告、B&C 商业合规 PCI 报告、度量报告、风险漏洞数据库报告、Tickets 状态报告、用户活动报告以及漏洞报告等方面，输出 PDF 形式如图 9-136 所示。

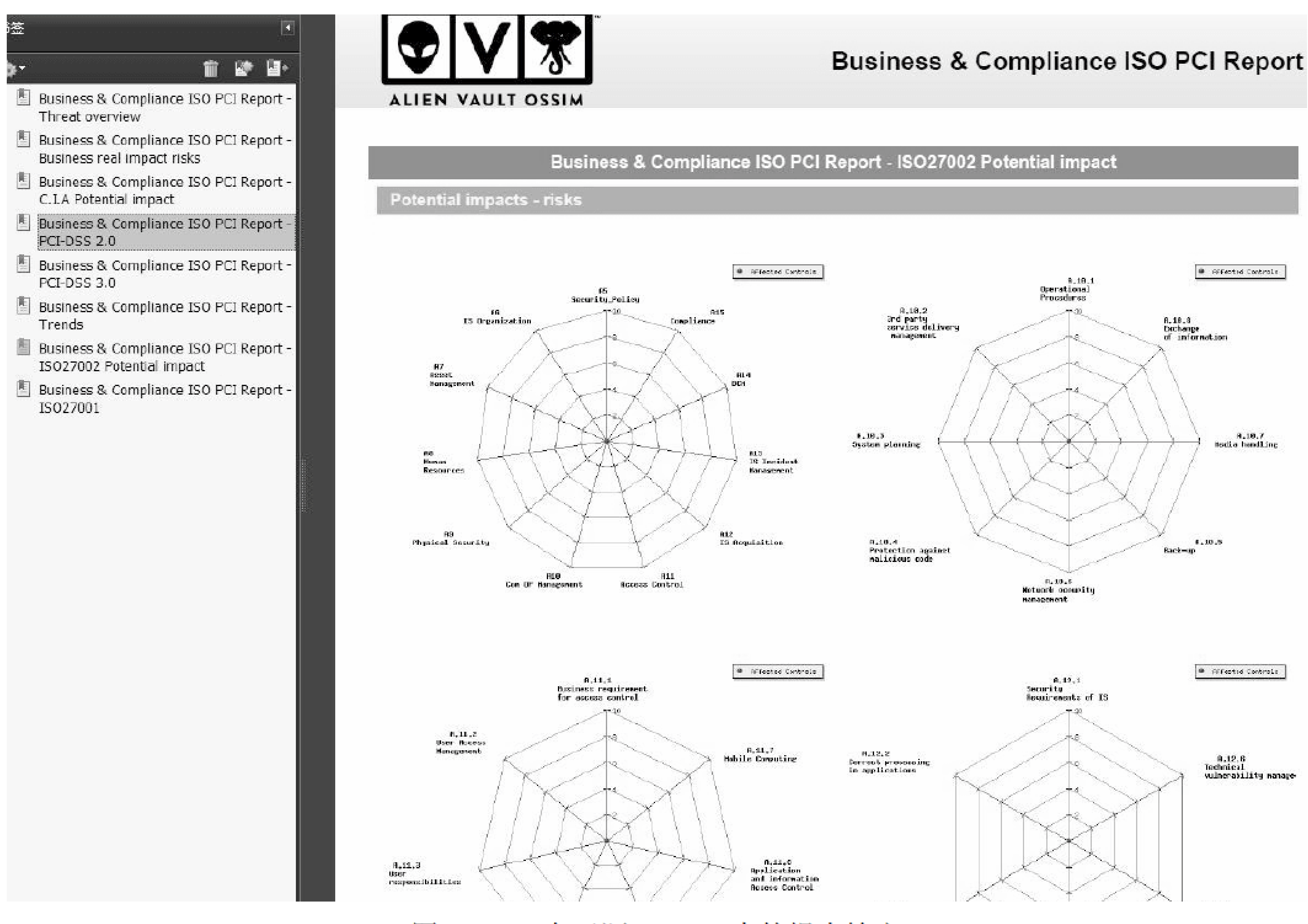


图 9-136　在开源 OSSIM 中的报表输出

9.13.6　日志合规检测

企业网中存在大量不同种类的日志，而且数据量巨大，使得管理员需要花费更多的精力去收集和分析日志。目前国际上对日志的管理都有这一套严格的标准和完善的依从性技术，法律法规（PCIDSS、FISMA、HIPAA）以及最佳实践框架（如 ISO 27001 等），这些规定都对日志管理提出了强制性的要求，希望读者在收集日志的同时对这些标准有所了解。

ISO 27001 环境下记录的日志内容为 A.5.1（Information Security Policy）~A.15.3（Information Systems Audit Considerations），所要求的详细内容被记录在“ISO/IEC 27001:2005 信息技术-安全技术-信息安全管理系统-需求”中，如果所在的组织正在推行 ISO 27001，可到 http://www.27000.org 查询相关信息。在 OSSIM 中对 ISO27001 也进行了严格的定义，大家可以在 Configuration→Threat Intelligence→Compliance Mapping 菜单下找到，默认系统为英文，为了方便理解汉化内容大家可参考 http://chenguang.blog.51cto.com/350944/1671973 这篇博文。另外，对于报表的检测模块在 OSSIM USM 中还能够自定义输出内容，如图 9-137 所示 ，图中左侧为选择好的报表模块，也可以将右侧待选模块加入其中。

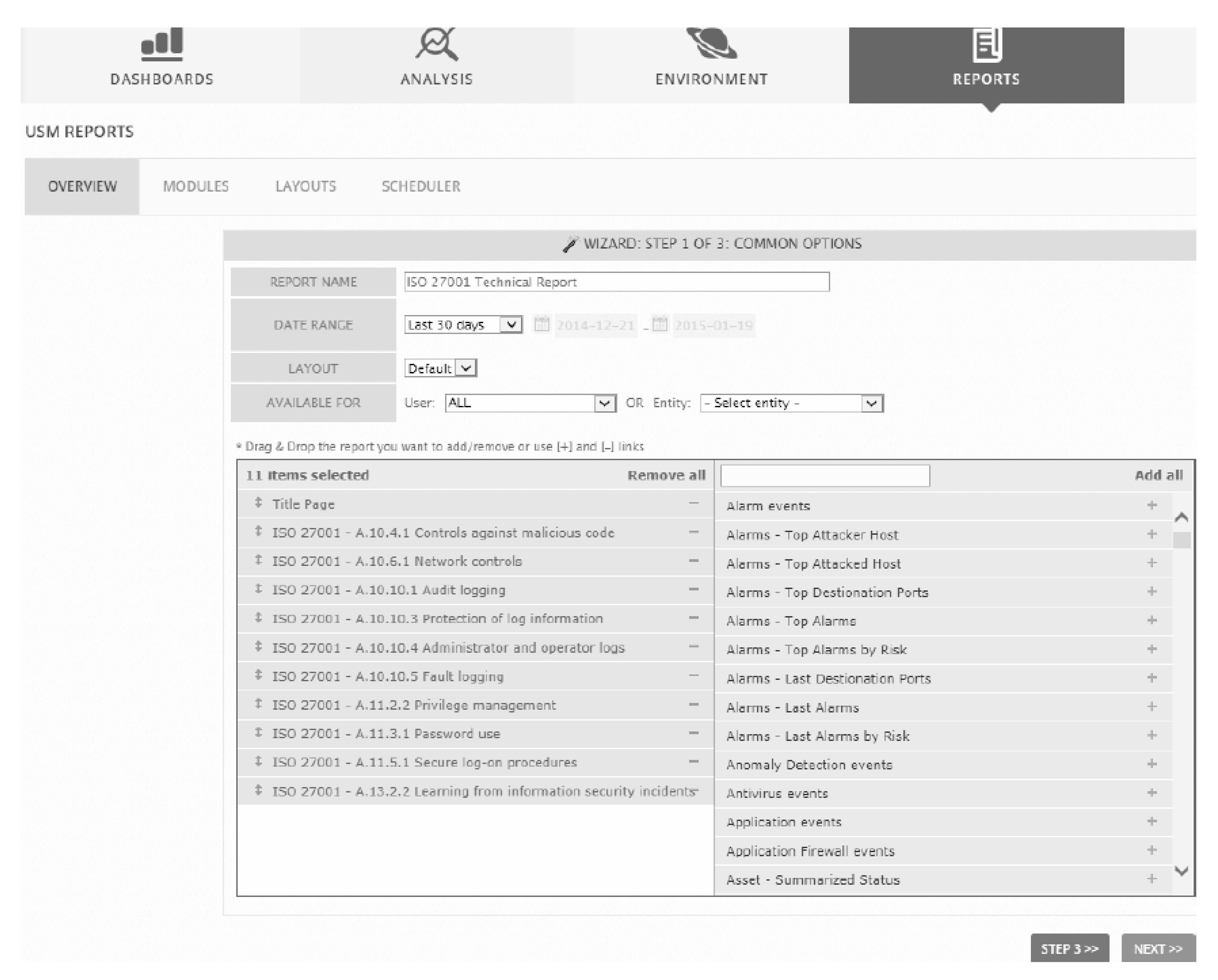


图 9-137　自定义报表

模块选定之后，接着确定对那些资产进行审计，如图 9-138 所示。

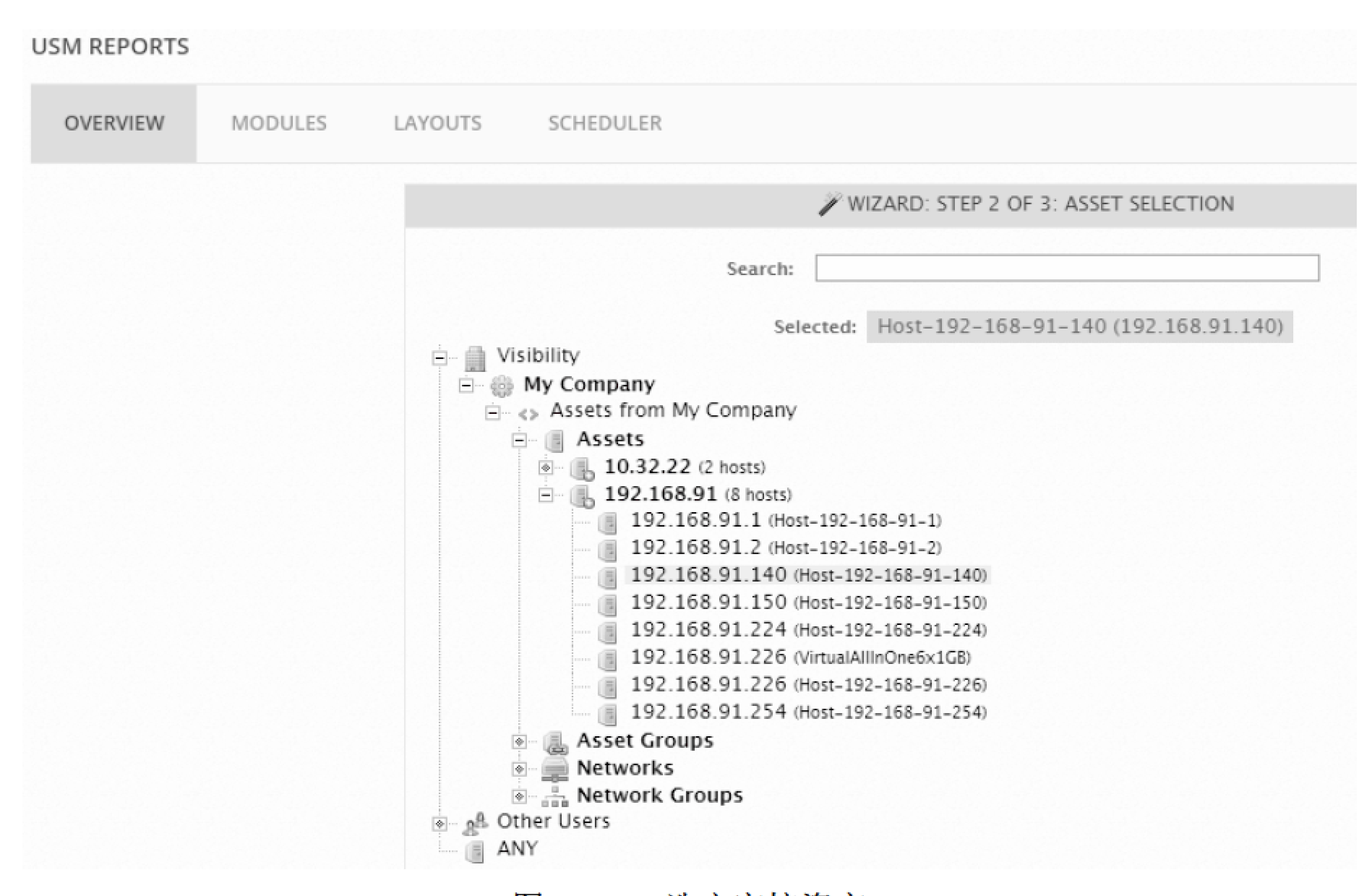


图 9-138　选定审核资产

待目标选定之后，接下来指定报表的 Logo、主题内容、地址、电话、生成日期等内容，如图 9-139 所示。

MODULES　LAYOUTS　SCHEDULER

WIZARD: STEP 3 OF 3: CUSTOM PARAMETERS FOR SELECTED REPORT MODULES

TITLE PAGE

Logo: D:\Users\dell\Desktop\noc.jpg 浏览...

* Only .gif, .png and .jpg files

Main Title: ISO 27001 Technical Report

I.T. Security:

Address:

Tel: 010-12345678

Date: #DATE

Add a custom comment

ISO 27001

A.10.4.1 CONTROLS AGAINST MALICIOUS CODE　ADD AS A NEW REPORT MODULE

Number of Events: 25 (25, 137, 250, 1,000, 10,000)

DS Groups: Directive events

Source Database: SIEM

Group by: Sensor

Add a custom comment

图 9-139　生成报表格式

一切就绪之后，单击“更新”和“运行”按钮，系统立即快速生成定制好的报表。默认检测时间为 30 天，系统最长可以产生 1 年的报表，如果需要调整时间，则需要选择自定义运行报表，如图 9-140 所示。这里可以选择的时间范围从当天到全年都可以自由选择，按运行按钮即可输出报表，如图 9-141 所示。

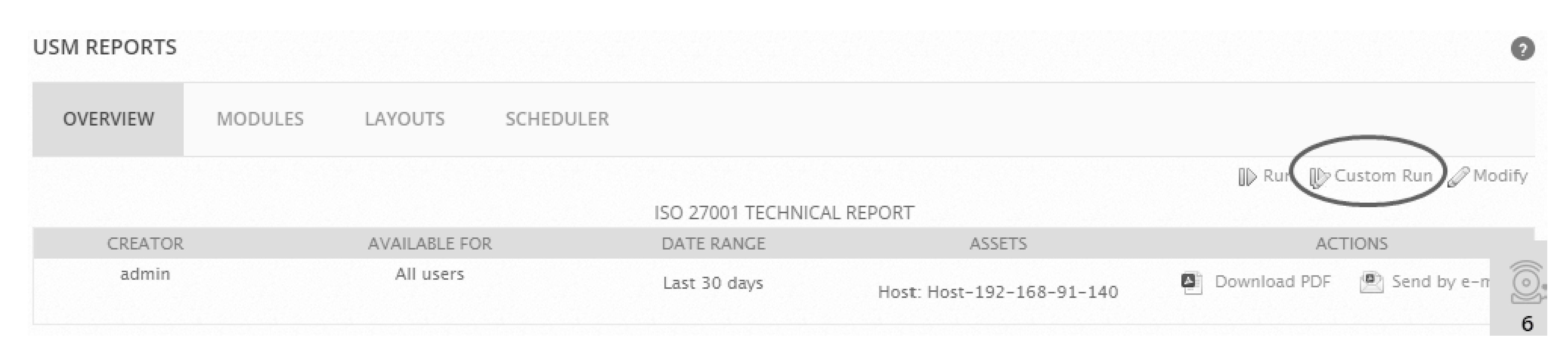


图 9-140　自定义运行

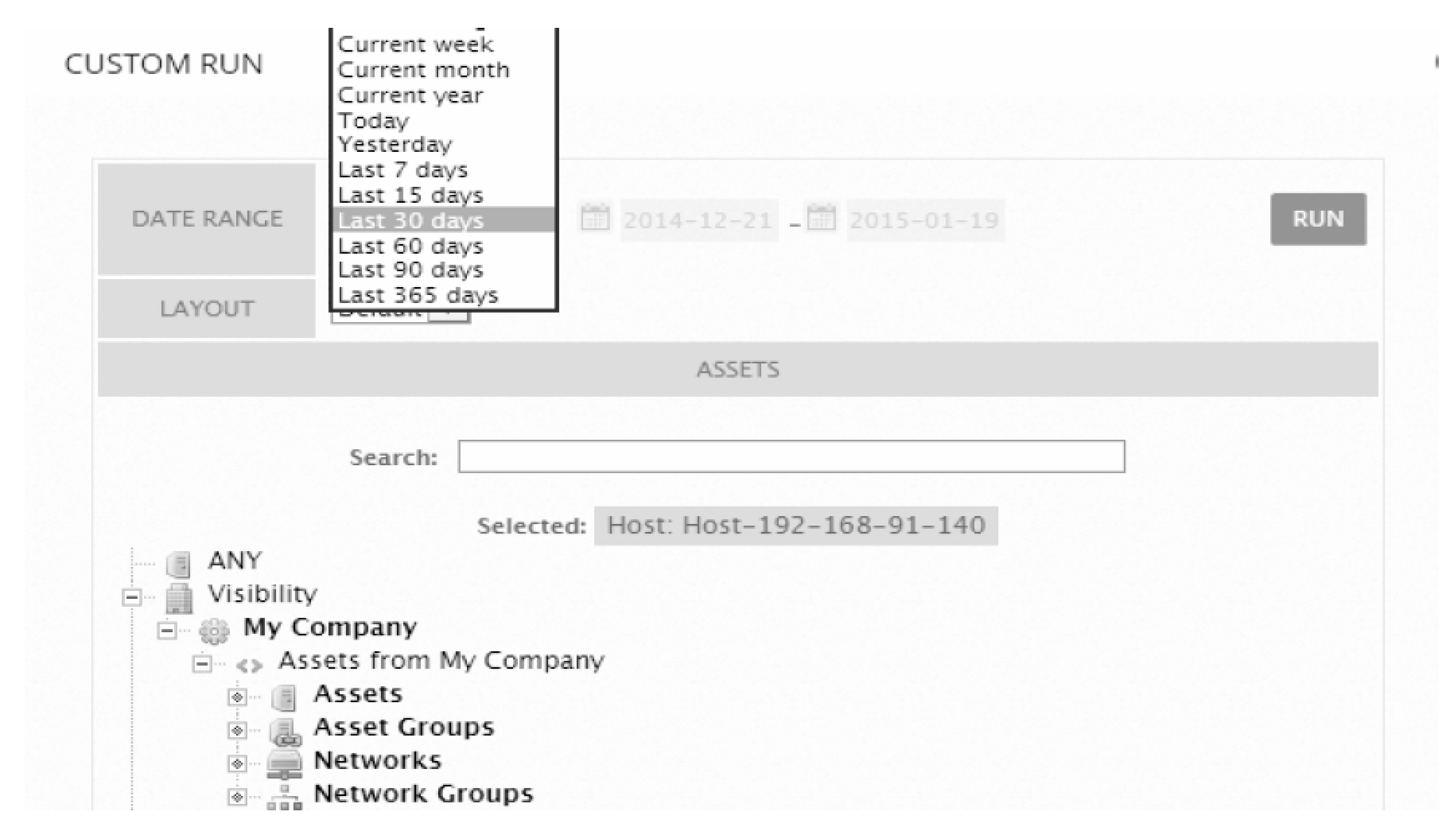


图 9-141　选择时间范围

9.13.7　报表合规性

为了解合规性，下面我们先来了解几种流行的法规，查看它们在日志记录和分析管理中扮演的角色。这些报表输出位于 Reports 菜单下。在 OSSIM 系统中合规性类别分为以下 4 种类型。

1. PCI——支付卡行业数据安全标准

OSSIM 系统中，通过以下路径访问 PCI DSS 的内容，Web UI 下 Configuration→Threat Intelligence→Compliance Mapping，如图 9-142 所示。

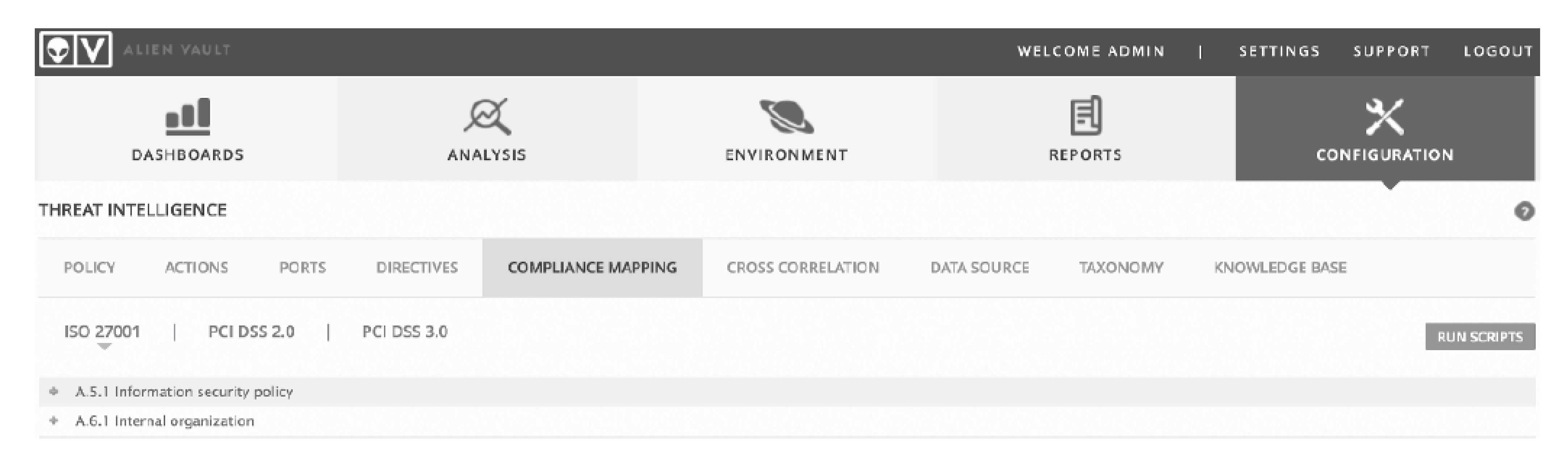


图 9-142　合规显示

它可使企业满足 PCI-DSS（支付卡行业数据安全标准）中第 10 条款的要求。该条款要求支付服务提供商必须记录系统日志，以便追踪并报告对其网络资源和持卡人数据的所有访问。

网络环境中的日志，便于出现问题时用于计算机取证分析。如果没有系统活动日志，则很难找到故障原因。OSSIM 并不是简单将 ISO 27001、PCIDSS 2.0 等规章进行罗列，它会在 Report 菜单中将合规要求以报表形式输出。表 9-7 中展示了 PCI 报表输出表的动作及描述。

表 9-7　PCI 报表输出主要内容

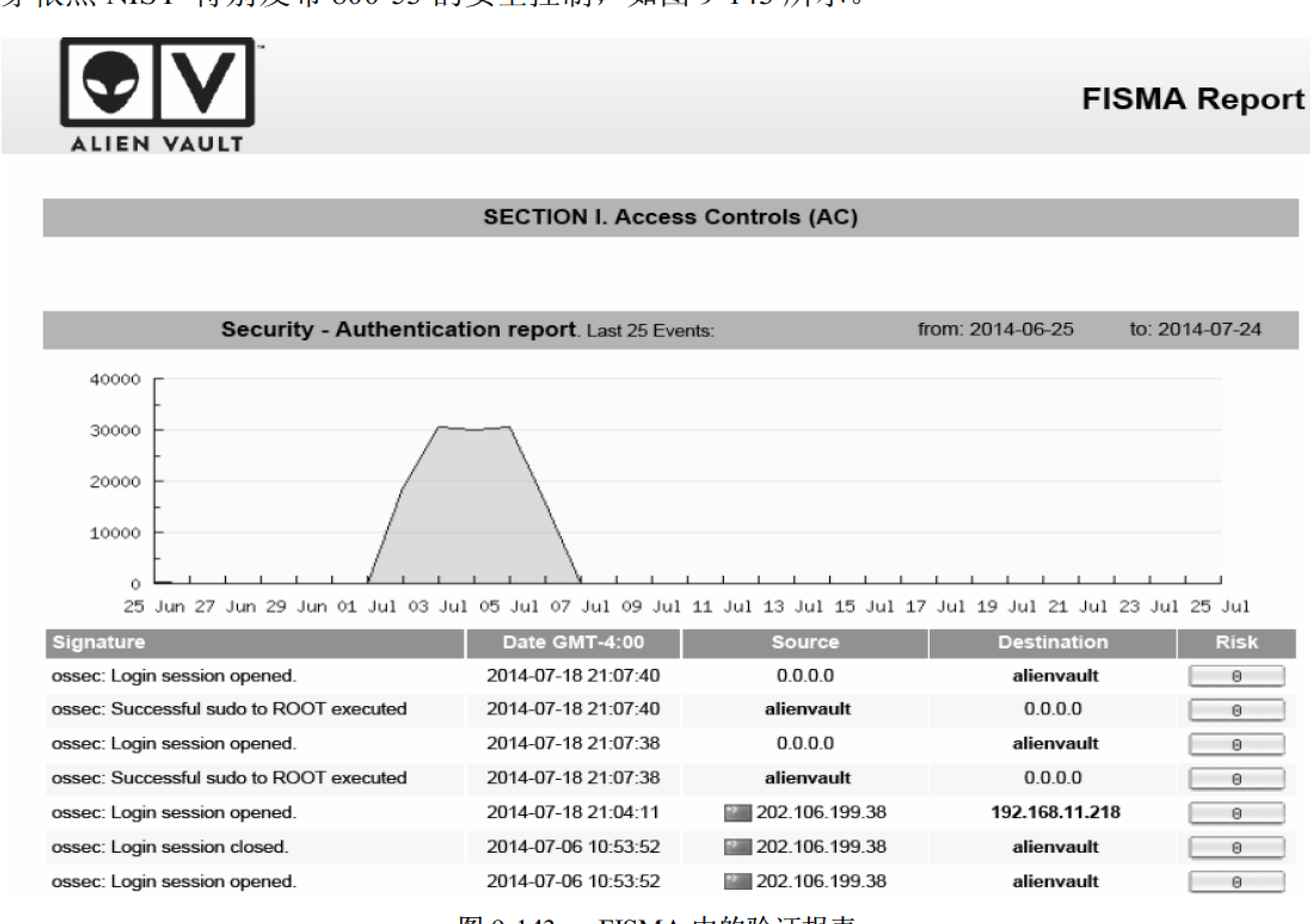
表名	动作	描述	组 成 表
对象访问	PCI-DSS 的 10.2.7 条款	系统级别对象的创建和删除，鉴别出某特定对象（文件、目录等）何时被访问，访问类型（如：读、写、删除）以及访问是成功或失败，以及谁执行了访问动作等	批准访问一个已经存在的对象类型来访问一个对象 批准访问一个已经存在的对象类型来创建一个对象。 批准访问一个已经存在的对象类型来修改一个对象。 一个要打开对象并删除对象的尝试。 一个保护的对象被删除
登录	PCI-DSS 的 10.2.1 和 10.2.3 条款	清晰地指明，为了防止滥用，所有个体用户对系统的访问都要被记录和监视。其目的不只是为了抓住黑客，而是要记录下所有对持卡人数据的访问。大多数情况下，对访问进行记录就足以制止恶意活动，就很像停车场中的摄像头。对系统的非法逻辑访问企图也将被记录和监视，从而可以有效防止数据的乱用。因此，在这类报表中，必须提供用户名、日期和时间等信息	用户成功登录到计算机。 成功访问了系统。这个事件指明一个远程用户已经成功从网络中连接到服务器上的本地资源，为网络用户生成了一个令牌。 用户退出一个计算机。 未知用户名或已知用户密码错误的登录尝试。 在允许的时间之外的登录尝试。 一个使用了停用/过期账户的登录尝试。 账户在登录尝试的时候被锁住。这个事件指明一个不成功的口令攻击导致了账户被锁住。 一个用户已经重新连接到断开的终端服务会话。这个日志指明一个以前的终端服务会话已经连接。 一个用户没有退出就断开了终端服务会话。这个事件发生于一个用户通过网络连接到终端服务会话的时候。出现于终端服务器
策略更改	PCI-DSS 的 10.2.3 条款	所有审计记录的访问，让组织通过追踪事件日志，了解任何安全审计策略的更改，从而更好地实现内部控制	一个用户权利被分配/移除。 另外一个域的委托关系被创建/移除。 一个审计策略被更改
用户访问	PCI-DSS 的 10.1 和 10.2.2 条款	审核具有 root 权限或者管理员权限的用户的所有动作。如：通过建立链接将对系统部件的所有访问链接到个体用户的过程	所有特权用户安全相关的动作
系统事件	PCI-DSS 的 10.2.6 条款	审计日志的初始化，要求定期复查信息系统的活动记录（如：审计日志）	计算机的重启是否为计划中的重启，获取一个计划外的重启。 一个可靠的登录进程已经注册到本地安全认证。修改或清除安全日志

在 OSSIM 中列出的 PCI DSS2.0 的 12 类需求（汉化内容大家可访问 http://chenguang.blog.51cto.com/350944/1671930 这篇博文），这些需求覆盖了数据访问、特权用户操作、日志访问和初始化、失败和无效日志的访问、身份认证及授权策略以及对可疑活动的

监视等。目前 PCI 安全标准委员会已发布支付卡行业数据安全标准 3.0 版（PCI DSS 3.0），主要增加了对 PCI DSS 范围内的系统组件进行库存管理。通过学习 OSSIM 系统中 ISO 27001 和 PCI DSS 相关内容，不仅可改变用户不良安全习惯，而且增强了对落实遵从管理的执行力。

2. FISMA——联邦信息安全管理

该法案旨在规定联邦信息和信息系统的最小安全控制要求。它指定了联邦信息和信息系统在 17 个安全相关的领域的最小安全要求。联邦部门必须满足这里定义的最小安全要求，并贯穿依照 NIST 特别发布 800-53 的安全控制，如图 9-143 所示。



Signature	Date GMT-4:00	Source	Destination	Risk
ossec: Login session opened.	2014-07-18 21:07:40	0.0.0.0	alienvault	0
ossec: Successful sudo to ROOT executed	2014-07-18 21:07:40	alienvault	0.0.0.0	0
ossec: Login session opened.	2014-07-18 21:07:38	0.0.0.0	alienvault	0
ossec: Successful sudo to ROOT executed	2014-07-18 21:07:38	alienvault	0.0.0.0	0
ossec: Login session opened.	2014-07-18 21:04:11	202.106.199.38	192.168.11.218	0
ossec: Login session closed.	2014-07-06 10:53:52	202.106.199.38	alienvault	0
ossec: Login session opened.	2014-07-06 10:53:52	202.106.199.38	alienvault	0

图 9-143　FISMA 中的验证报表

FISMA，本身没有规定任何日志记录或日志管理，它只是停留在策略和规划层级，例如 AC 指的是访问控制，但这里只有文档却没有行动，没有告诉如何去做。

AU 控制，代表审计和可审核性策略及规程，这里最重要的是要有日志记录策略，如果没有作为基础的操作规范就没有任何意义，要定义所有记录的事件，并在每个事件所生成的细节上做记录，有一点可以肯定，这些规定、文档并不能阻止黑客，而且当日志数据丢失时，那么将无助于对攻击事件的调查。

3. HIPAA -医疗电子交换

HIPAA 法案要求对所有日志（包括操作系统和应用程序的日志）进行分析。HIPPA 和 PCI

DSS 不同，HIPPA 本身没有涉及安全控制级别和所采用的技术，所以这种报表重要程度较低，输出效果如图 9-144 所示。

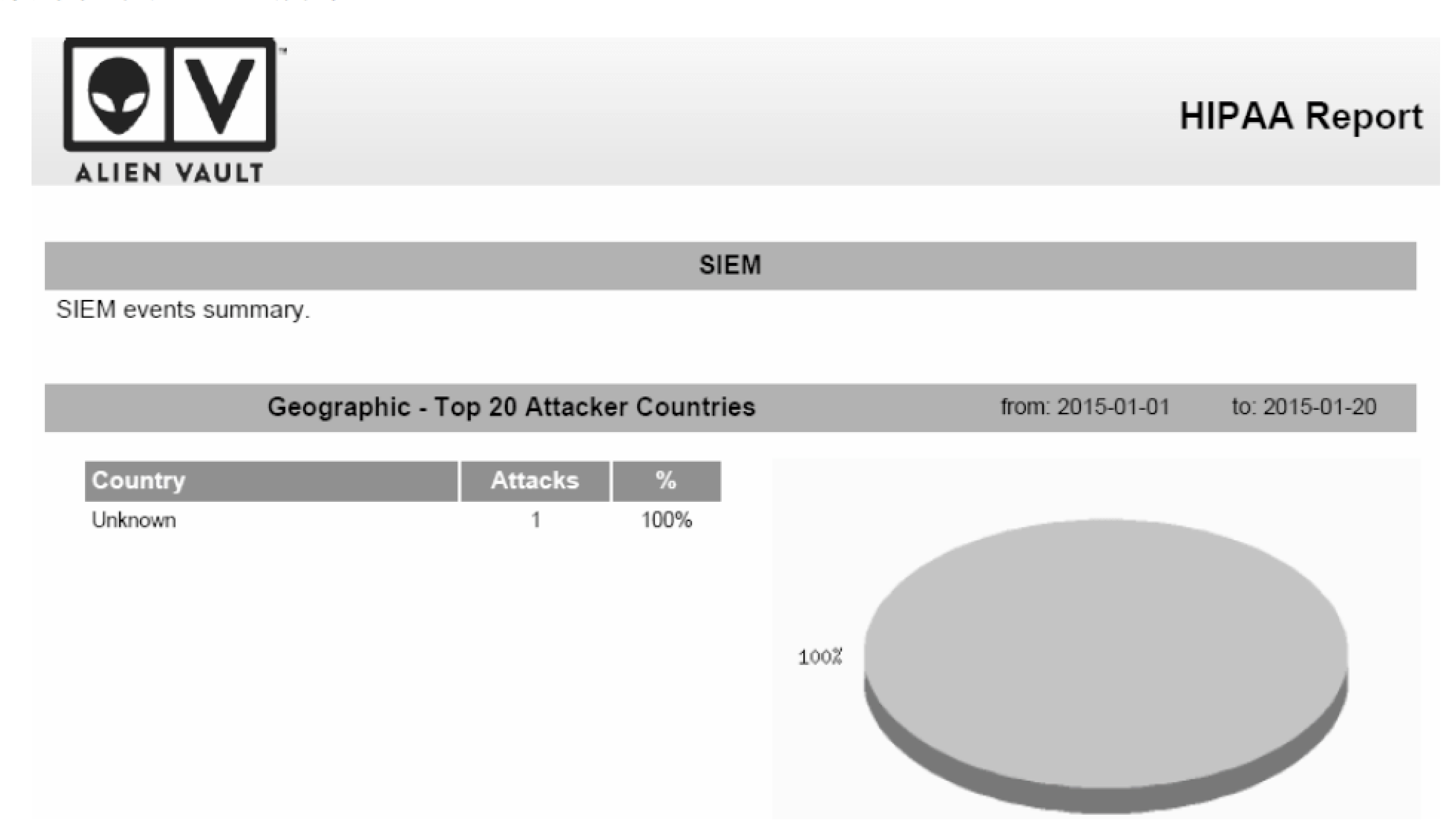


图 9-144　HIPAA 输出报表

9.14 小结

学习本章需要结合前面 8 章的内容，从 Web UI 结构讲起，逐步介绍 HIDS 的安装以及规则的编写初步、深入阐述了 OpenVAS 原理与故障处置，对于本章资产管理也是个重点内容，从资产中的代理安装到异常流量监控到漏洞扫描都有详细介绍，尤其对网络蠕虫的防范进行了详细地讲解。本章最后一部分讲解了合规管理和统一报表管理，对报表的基本使用做了详细描述。

第 10 章

◀ 基于B/S架构的数据包捕获分析 ▶

抓包是运维的必备技能，很多网络故障需要靠抓包来解决，如常见的 ARP 欺骗和广播风暴。另外还有一些网线或光纤接触不好的故障，不抓包也很难分析出来，例如两个公司之间互联，网线测试都没问题，但始终不通。经过抓包分析表明，发现其他单位的 ping 请求都伴随着 ARP 查询，而不走路由，这时怀疑有可能掩码设置错误的问题，经仔细排查，确实是路由器上的掩码出现失误。抓包工具有不少，但选择一款适合你的工具非常重要。

本章主要为大家介绍 OSSIM 中故障排除利器——基于 Web 的数据包分析工具，它是 Wireshark 的另一种表现形式，这和 CloudShark Appliance（cloudshark.org）的表现手法非常类似。从功能上看，这种基于 Web 的抓包分析工具是 Wireshark 的精简版。它的优势在于远程客户端，通过 Web 界面就可以实现在服务器端抓包，还能够抓到不同传感器（能收集不同网段的信息）所检测的数据包异常。在阅读本章时，需要读者具备网络嗅探与数据报分析的基础知识，具备 Wireshark 抓包经历。

10.1 数据包捕获

过去，分析网络故障常在一个系统中用 tcpdump 抓包，将包保存起来，再导入 Wireshark 进行分析。不过现在使用 OSSIM Web UI 下的 Traffic Capture 不必这么麻烦，操作方法为 Environment→Traffic Capture，界面如图 10-1 所示。如果是在分布式 OSSIM 环境中，一定要注意 Sensor 的选择，以及 Sensor 上对应的网卡。

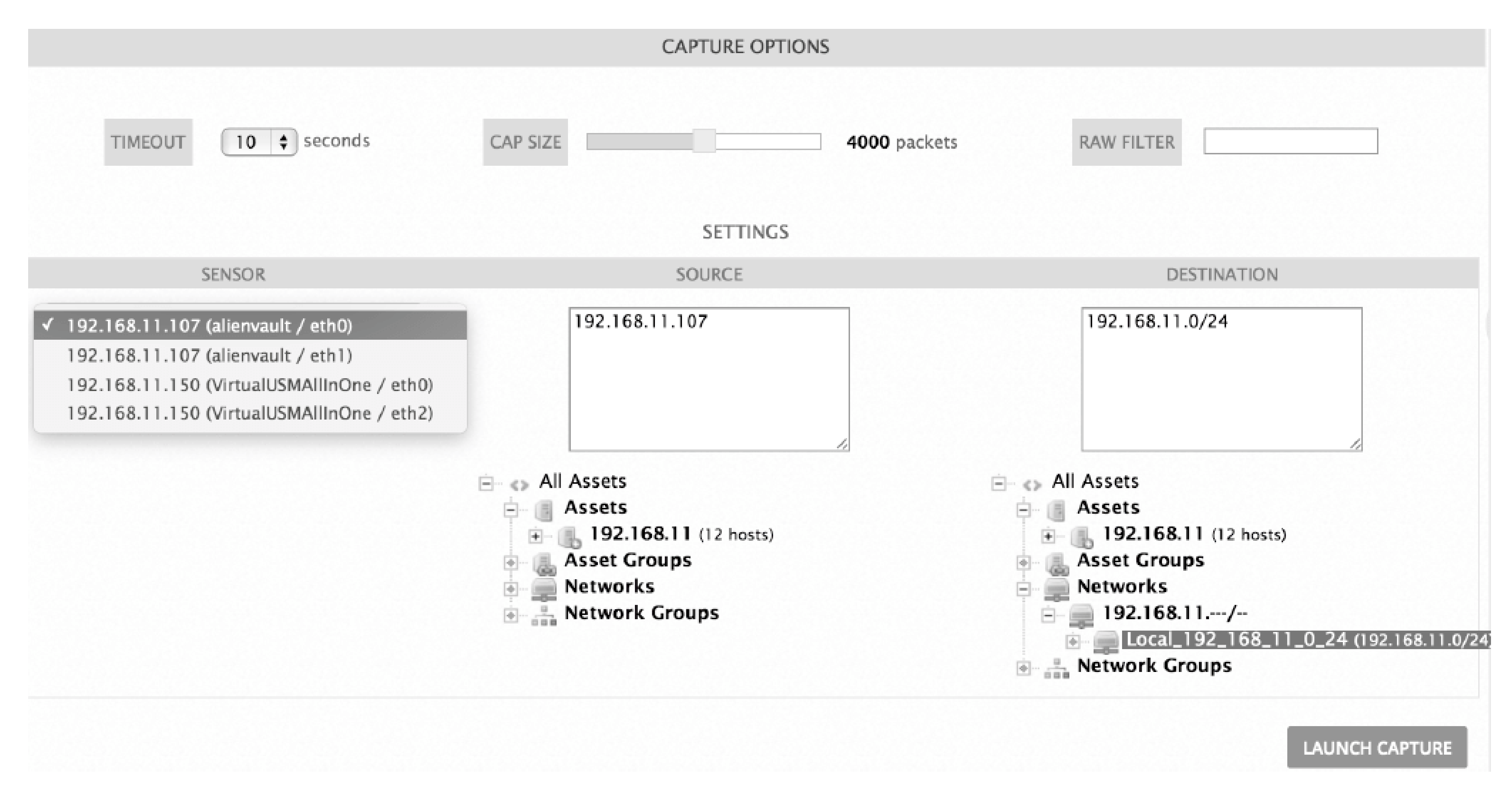


图 10-1　多传感器抓包设置

在设置（settings）选项下方，源地址和目标地址均为可选项。其使用方法较简单，选择对应 Sensor，输入源地址和目标地址，然后单击捕捉按钮即可。但应用该工具之前，操作人员对 TCP、UDP、IP、ICMP 协议及 HTTP、DHCP、DNS、FTP 等常见应用层协议需要熟悉。

单击捕捉按钮之后，会显示如图 10-2 所示界面。

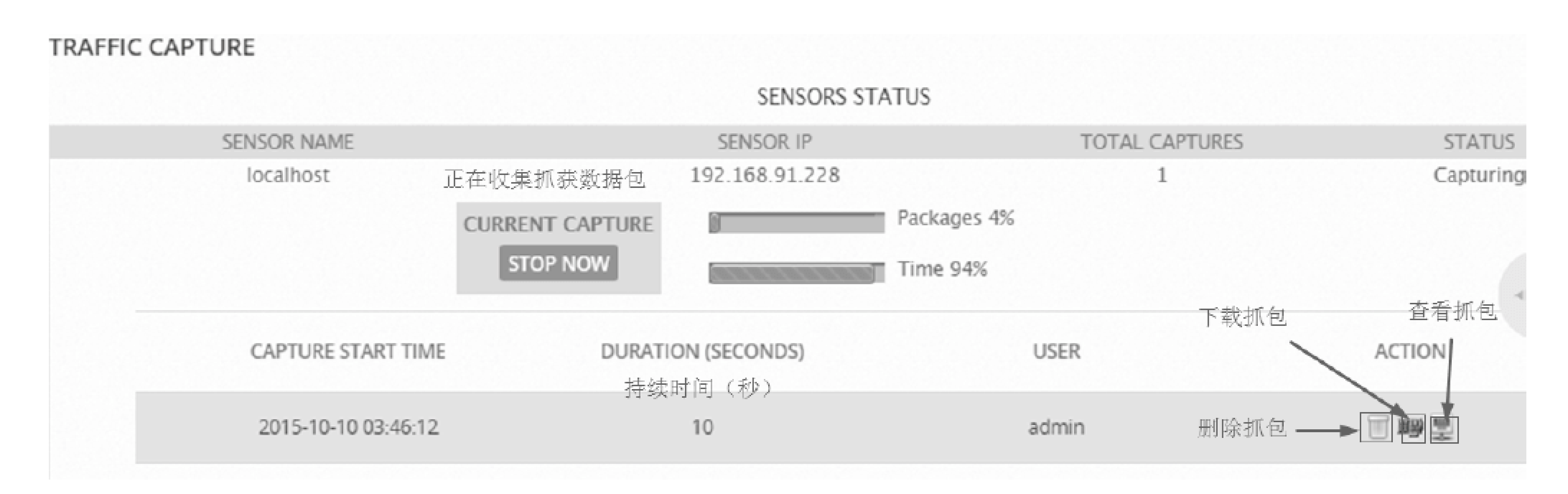


图 10-2　显示当前传感器抓包情况

另外，在分布式 OSSIM 系统中，分析远程某个网段的数据包，由受控网段内的 Sensor 抓包文件存储在 Sensor 端/var/ossim/traffic/目录下，命名为 netscan_admin_IP.pcap，前提是这台 Sensor 的/var 目录剩余空间大于 5GB。可以对所抓包列表进行删除，下载和显示载荷操作，如图 10-2 所示。

10.1.1　数据包捕获设定

在设定数据包时，可以在 Sensor 下拉菜单中选择需要监控网段的传感器，选定网络接口，

以及捕包时间（10~180 秒），选择源地址、目标地址，设定过滤选项（Raw Filter）以及设定每个捕获数据包的大小等几个步骤。

Raw filter 可设置过滤协议，例如过滤 TCP、IP、UDP 等协议，其可选值为 tcp、upd、ip、icmp、arp、rarp 等。如果不填写，则代表所有协议。很多过滤条件需要手工输入，另外填写在 Raw filter 后面的一定是协议名称的小写字母。捕捉到数据包以后，下面开始查看细节，如图 10-3 所示。

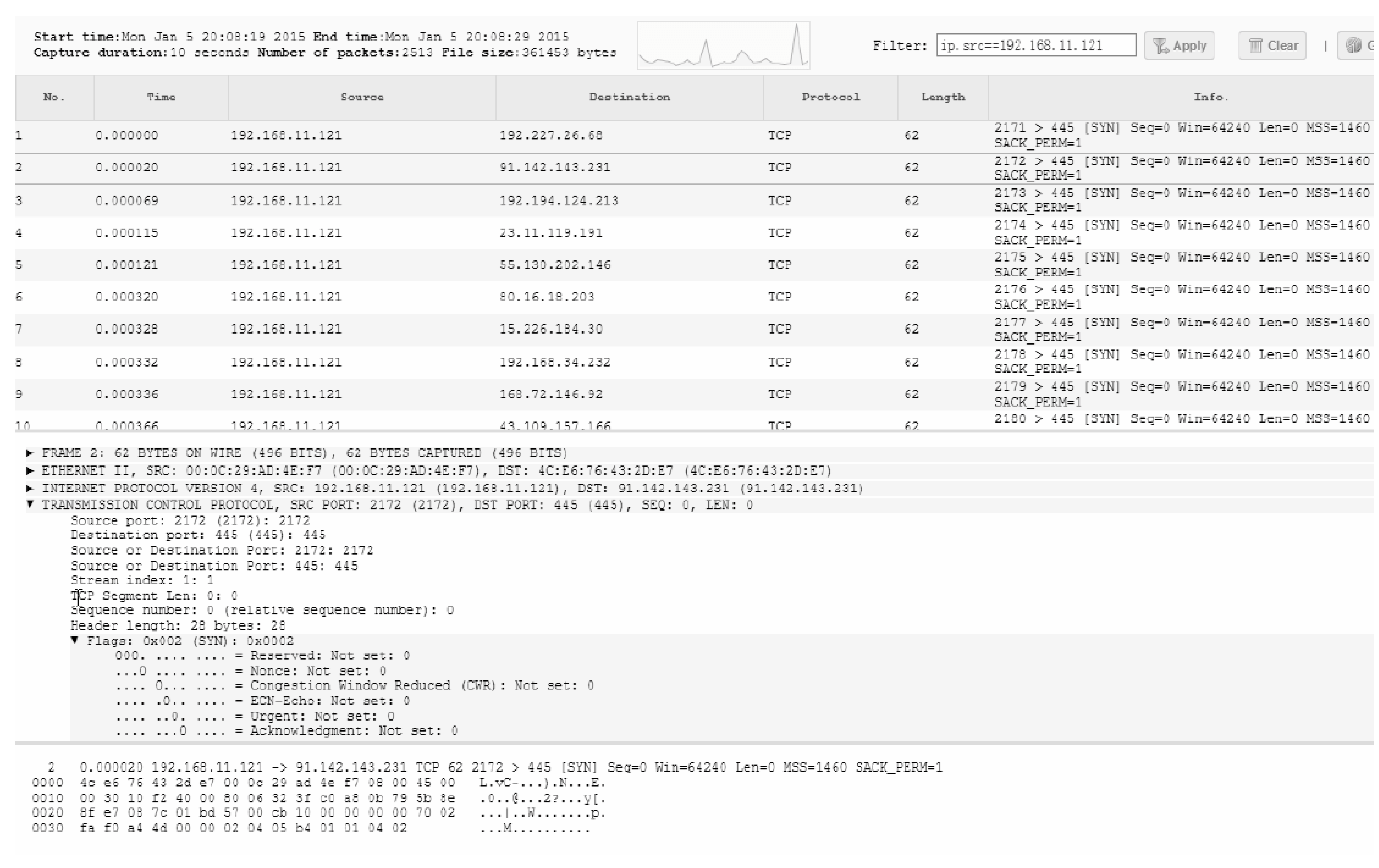


图 10-3　数据包细节

上图看到的是基于 Web 抓包系统的主窗口，外观和 Wireshark 主界面非常相似。最上方显示捕获数据包的开始时间、结束时间和持续时间，在主窗口下方有 3 个可伸展的面板，分别是包列表、包细节和包字节 3 部分。如果没抓到包有可能是流量镜像环节有问题，也有可能是过滤条件设置不当所致。

10.1.2　抓包区域说明

下面介绍图 10-3 所示数据包中各栏作用如下：

（1）包列表。最上面的面板显示了当前捕获文件中的所有数据包，包括了数据包序号（NO.）、数据包捕获的相对时间（Time）、源地址（Source）、目的地址（Destination）、协议（Protocol）、数据包长度（Length）以及概况信息等。为了便于观察，对于选中的包，系统会用绿色高亮显示出来。

（2）包细节。中间面板显示了一个数据包中的内容，并且可以通过展开或者收缩来显示这个数据包中所捕获到的全部信息。

（3）包字节。最下面的面板可能是比较头痛的，如图 10-4 所示，因为它显示了一个数据包的真实面目，用过十六进制编辑的读者就能理解，图中左边方框内显示的是偏移量，中间显示的是十六进制，右边方框显示的是中间内容的可显示字符。

```
0000  14 10 9f cd f6 ef 00 0c 29 12 ff d6 08 00 45 00   ........).....E.
0010  00 54 84 55 00 00 40 01 5e 9b c0 a8 0b 64 c0 a8   .T.U..@.^....d..
0020  0b 04 00 00 78 f0 38 25 03 93 52 f1 c6 18 00 00   ....x.8%..R.....
0030  47 4a 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   GJ..............
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37                                             67
```

图 10-4　十六进制显示包内容

10.1.3　抓包时提示“This traffic capture is empty”的解决办法

当抓包之后，需查看结果，此时我们单击 View Payload （“View Payload”），系统提示“This traffic capture is empty”，这说明没抓到有效数据包，我们需要调整“Capture Options”数据包选项，在“Cap Size”选项中默认为 4000 packets，这时试着将滑块向右移动，例如 100~200 之间，并延长抓包时间，与此同时在界面中把 RAW Filter 选项后面清空。

10.1.4　远程故障排除案例

当网络出现故障时，由于直接用 tcpdump 抓包分析有些困难，而且当网络中大量发送数据包时分析更加不容易，这时使用-w 参数，然后配置 Wireshark 分析效果更好。具体参数如下：

```
#tcpdump -i eth0 -c 2000 -w eth0.cap
```

- -c 2000：表示数据包的个数。
- -w：表示保存 cap 文件，方便用其他程序分析。

在本地机房维护时，如遇到网络病毒爆发或者通信故障，管理员会通过上面抓包方式来分析故障，但如果是异地机房出了同样的问题，你需要在远程深度分析网络数据包的情况，该怎么办？你可能需要临时联系对方运维人员开始配置端口镜像，并部署抓包设备，而这种低效的响应方式往往耽误了分析故障的最佳时机。

下面我们看看通过分布式 OSSIM 部署的解决方法，公司总部 C 部署一台高性能 OSSIM 服务器，在异地分公司 A 机房和 B 机房部署了 Sensor，它们通过 VPN 和 Ossim Server 连接。其拓扑如图 10-5 所示，异地机房的 Sensor 分别连入交换机的 SPAN 口，当工作需要时，能够实时抓取网络数据。

如果想获取网卡 etho 中除了 TCP 22 端之外的所有流量，输入以下命令：

```
# tcpdump -nni eth0 'not (tcp and port 22) '
```

还有一些过滤方法参见 10.2 节。

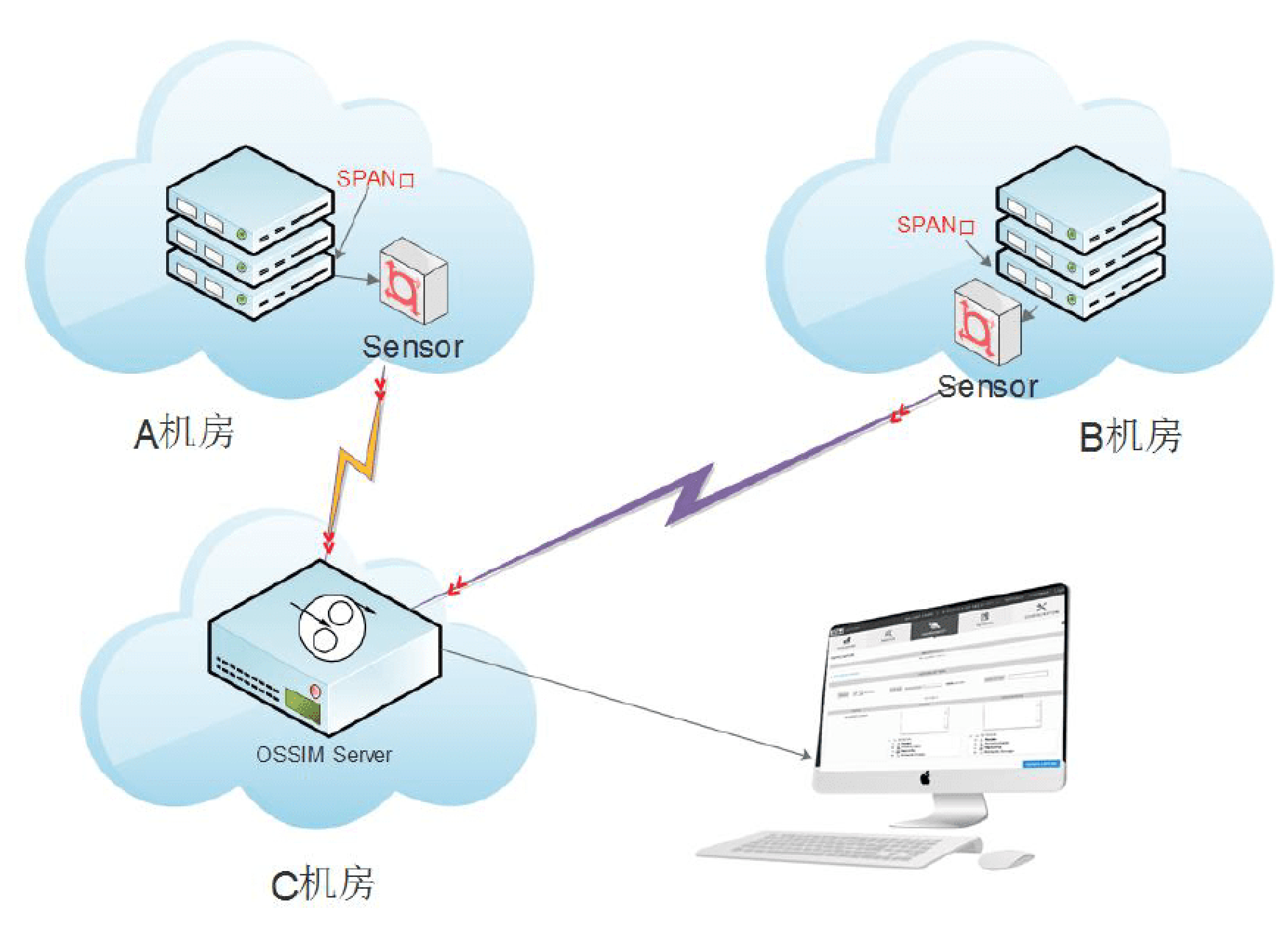


图 10-5　远程数据包分析

当 A 机房内服务器出现故障，可以连接到 OSSIM 的 Web UI 界面，通过 Traffic capture 并选择 A 机房内 Sensor，然后开始抓包，并进行后续分析，这样的部署大大提高了远程机房之间数据包分析的效率。

10.2 数据包过滤种类

收集到大量网络数据之后，需要对其进行一定的处理，也就是过滤，这种过滤主要分为以下 4 种：

- 基于主机（IP）的流量过滤，如果需要监控的信息中包含某个 IP，那么就可以采用这种基于主机的过滤方式。
- 基于端口的流量过滤，如果需要的信息中包含特定端口（源端口和目的端口），那么就可以用这种方式。
- 基于协议+端口的流量过滤。
- 基于主机+端口的流量过滤。

大家先了解以下几种操作方法：

（1）tcp dst port 22 作用：显示目的 TCP 端口为 22 的数据包。

（2）ip src host 192.168.150.10 作用：显示来源 IP 地址为 192.168.150.10 的数据包。

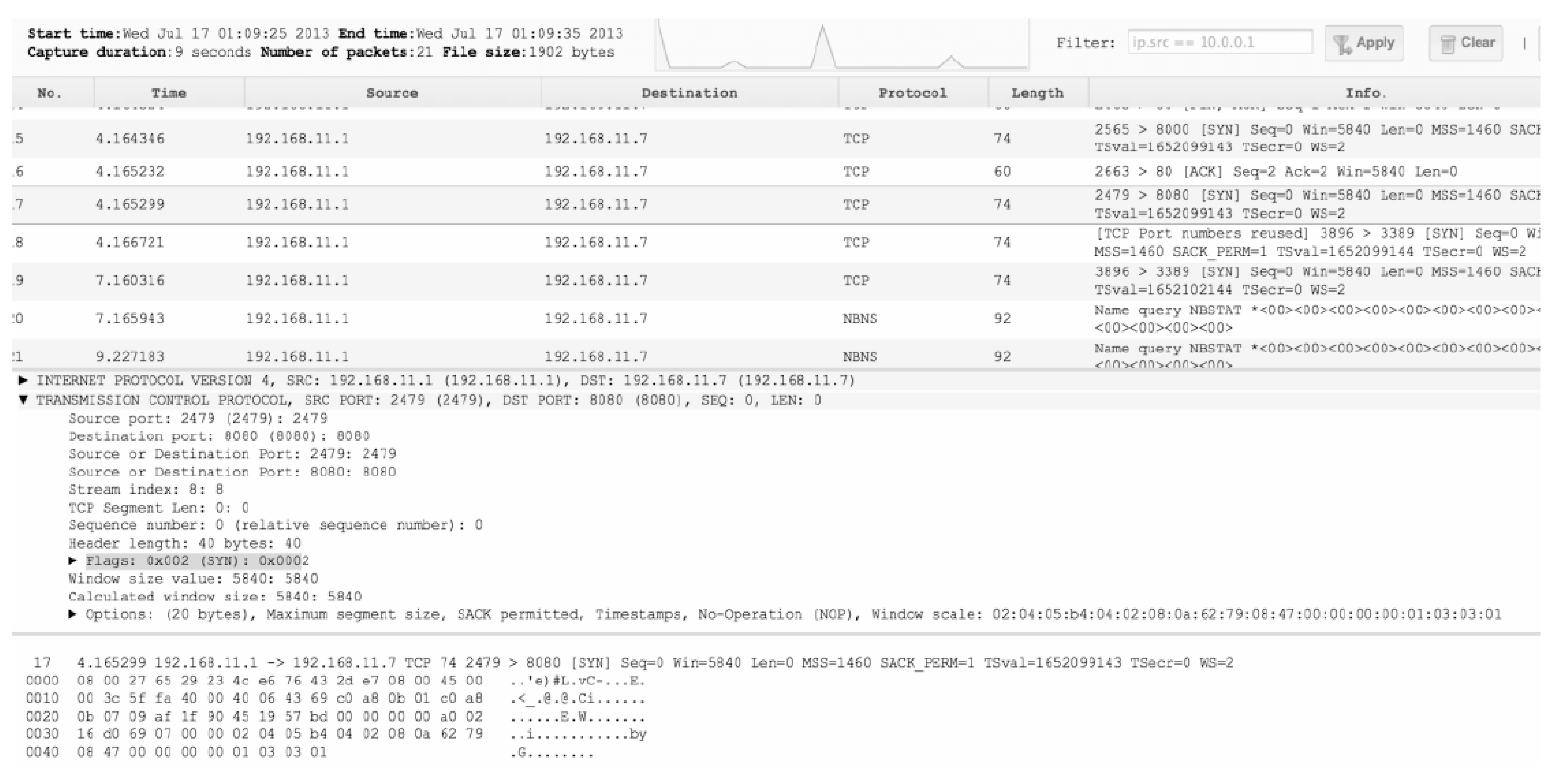
（3）host 192.168.150.10 作用：显示目的或来源 IP 地址为 192.168.150.10 的数据包。

（4）src port range 3000~5500 显示来源为 UDP 或 TCP，并且端口号在 3000~5500 范围内的数据包。

（5）not icmp 显示除了 ICMP 以外的所有数据包。

（6）src host 10.7.2.12 and not dst net 10.200.0.0/16 显示来源 IP 地址为 10.7.2.12，但目的地不是 10.200.0.0/16 的数据包。

用 Wireshark 打开 pcap 格式数据包后，每条消息中 field 会被解析出来，并按照协议层次折叠起来。第一层显示 Frame XXX，该级别没有对应某层具体的协议，而是对本条消息的概括性总结，描述了一些有用的概括性信息，比如从里面我们可以看到本条消息各种协议的层次关系，展开其他协议层之后，对应该协议的各个域，如果需要使用过滤时，可以使用以上介绍的几种方法，程序截获数据包分析如图 10-6 所示。如果单击 Graphs 按钮，可以获取到更多有关协议和流量的信息。



10-6　分析数据包

如果分布式 OSSIM 系统的 Sensor 出现问题，那么该功能将无法使用，下面看个例子，Sensor 和 Server 通信故障将影响到 Ntop、Openvas 以及抓包功能的使用。

如图 2-47 所示，此时在操作界面的左边方框中没有找到可用的 Sensor，状态描述为“×”代表不可用。这时抓包功能无法使用。

10.3 过滤匹配表达式实例

10.3.1 过滤基础

下面介绍几个最常用的关键字，“eq”和“==”等同，可以使用“and”表示并且，“or”表示或者。“!”和“not”则表示取反。对 IP 地址的过滤其中有几种情况：

（1）对源地址为 192.168.11.121 的包进行过滤，即抓取源地址满足要求的包。

表达式为：ip.src == 192.168.11.121

按照此方法我们可以填写到过滤的表单中，然后单击“Apply”按钮，即可过滤出结果。如果输入地址没有找到，则显示“no data with this filter”，则说明过滤条件有问题。

（2）对目的地址为 192.168.0.1 的包进行过滤，即抓取目的地址满足要求的包。

表达式为：ip.dst == 192.168.0.1

（3）抓取匹配源（或者目的）地址的 IP 地址是 192.168.0.1 的封包。

表达式为：ip.src == 192.168.0.1 or ip.dst == 192.168.0.1

10.3.2 协议过滤

（1）仅需要捕获某种协议的数据包，表达式很简单，把协议的名字输入即可。

表达式为：http

在进行数据包过滤时，在“Filter:”中输入 http 即可，如图 10-7 所示。

Start time:Mon Jan 5 20:08:19 2015 End time:Mon Jan 5 20:08:29 2015
Capture duration:10 seconds Number of packets:2513 File size:361453 bytes

Filter: http Apply Clear

No.	Time	Source	Destination	Protocol	Length	Info.
1536	5.932484	192.168.11.225	239.255.255.250	SSDP	139	M-SEARCH * HTTP/1.1
1540	5.944957	192.168.11.225	192.168.11.238	HTTP	311	GET /upnphost/udhisapi.dll?content=uuid:c3520d50-f800-a18c-1665305b469e HTTP/1.1
1546	5.953969	192.168.11.238	192.168.11.225	HTTP/XML	1339	HTTP/1.1 200 OK
1753	6.511530	192.168.11.238	192.168.11.225	SSDP	439	HTTP/1.1 200 OK
2234	8.691526	fe80::14ed:56ac:7818:3810	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
2310	8.938258	192.168.11.225	239.255.255.250	SSDP	139	M-SEARCH * HTTP/1.1
2329	9.054269	192.168.11.238	192.168.11.225	SSDP	439	HTTP/1.1 200 OK

▶ FRAME 1540: 311 BYTES ON WIRE (2488 BITS), 311 BYTES CAPTURED (2488 BITS)
▶ ETHERNET II, SRC: 00:26:C7:3B:D0:02 (00:26:C7:3B:D0:02), DST: 00:0C:29:D3:62:CC (00:0C:29:D3:62:CC)
▶ INTERNET PROTOCOL VERSION 4, SRC: 192.168.11.225 (192.168.11.225), DST: 192.168.11.238 (192.168.11.238)
▶ TRANSMISSION CONTROL PROTOCOL, SRC PORT: 54998 (54998), DST PORT: 2869 (2869), SEQ: 1, ACK: 1, LEN: 257
▼ HYPERTEXT TRANSFER PROTOCOL
▶ : GET /upnphost/udhisapi.dll?content=uuid:c3520d50-f800-40e5-a18c-1665305b469e HTTP/1.1\r\n

图 10-7 过滤 Http 协议

（2）排除某种协议的数据包。

表达式为：!tcp

排除 IP 协议：可以在“Filter:”中输入!ip 即可。

协议名称是小写字母。

10.3.3　对端口的过滤

（1）捕获某端口的数据包，如图 10-8 所示。

表达式为：tcp.port == 80（注意，80 前面为双等于号）

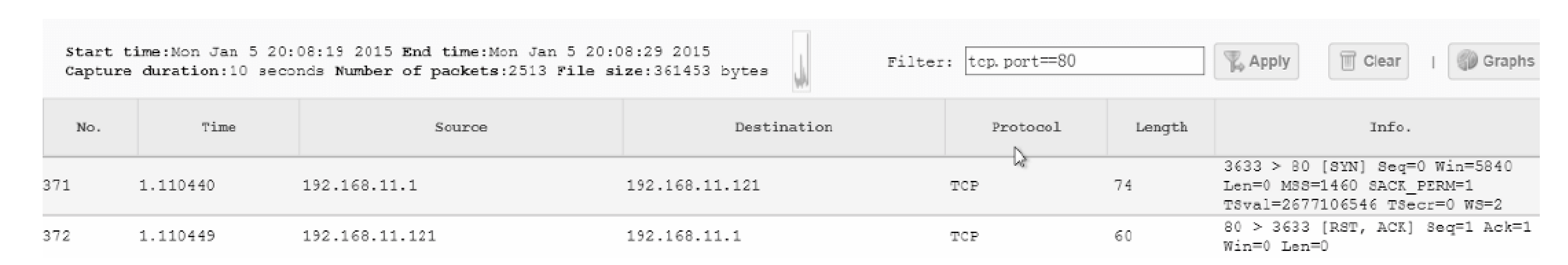


图 10-8　端口过滤

（2）捕获高位端口（大于 1024）的表达式：udp.port>= 2048

10.3.4　对包长度的过滤

分析一组数据包的大小，会发现很多问题。正常情况下，一个以太网的最大帧长为 1518 字节，去除以太网、IP 以及 TCP 头，还剩 1460 字节可供应用层协议的头或者数据使用。那么我们根据这个原则就能通过捕获数据包分析长度，了解其分布，然后对故障进行合理的分析，下面我们看看如何通过一些表达式多包进行过滤。

针对长度的过滤（这里的长度指定的是数据段的长度），如图 10-9 所示。

表达式为：udp.length< 86

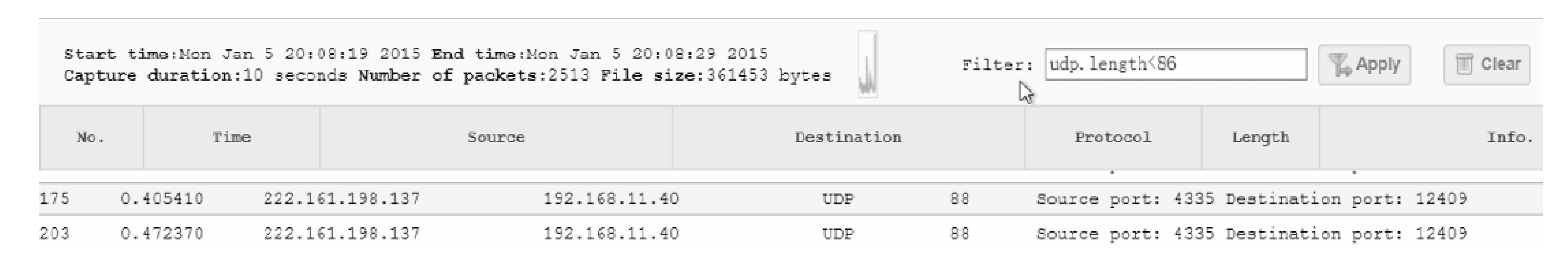


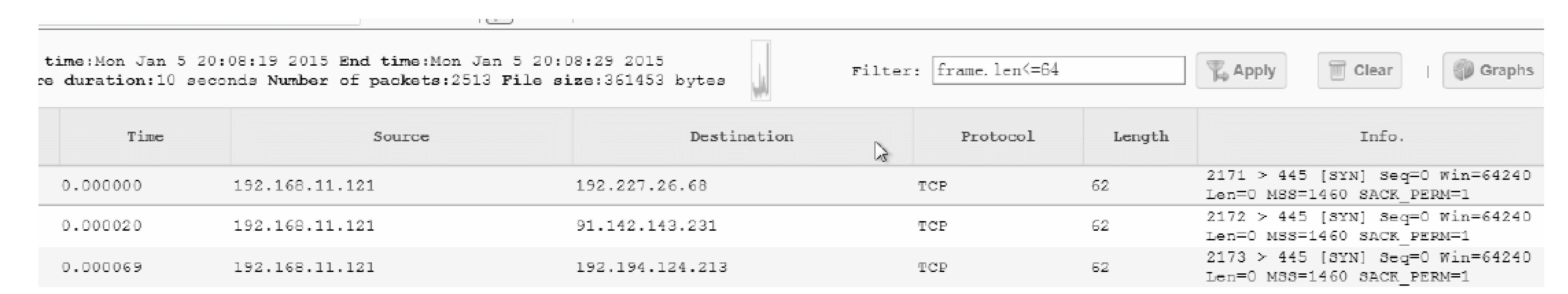
图 10-9　UDP 长度过滤

以太网头部为 14 字节（包括了 4 字节的 CRC 校验），IP 头最小 20 字节，没有数据以及选项的 TCP 数据包也是 20 字节，TCP 用于控制的数据包，例如 ACK、RST 以及 FIN 的大小大约是 54 字节。查看哪些长度小于 64 字节的数据包。

过滤器表达式为：frame.len<=64

效果如图 10-10 所示，同样方法可以使用如下表达式：

- frame.number<=10：表示过滤帧数小于 10 的包。
- frame.number= =10：表示过滤帧数为 10 的包。



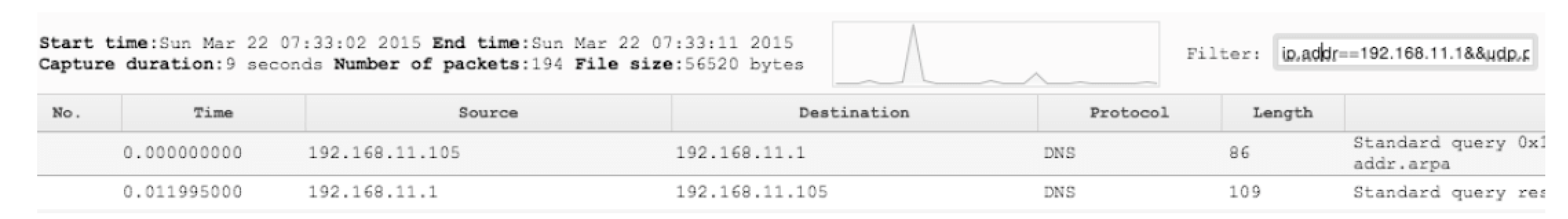
time:Mon Jan 5 20:08:19 2015 End time:Mon Jan 5 20:08:29 2015
re duration:10 seconds Number of packets:2513 File size:361453 bytes

Filter: frame.len<=64 | Apply | Clear | Graphs

Time	Source	Destination	Protocol	Length	Info.
0.000000	192.168.11.121	192.227.26.68	TCP	62	2171 > 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
0.000020	192.168.11.121	91.142.143.231	TCP	62	2172 > 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
0.000069	192.168.11.121	192.194.124.213	TCP	62	2173 > 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1

图 10-10　帧长度过滤

我们还可以根据 IP 地址＋端口进行过滤，比如，在“Filter:”后面输入“ip.addr==192.168.11.1&& udp.port==53”，效果如图 10-11 所示。



Start time:Sun Mar 22 07:33:02 2015 End time:Sun Mar 22 07:33:11 2015
Capture duration:9 seconds Number of packets:194 File size:56520 bytes

Filter: ip.addr==192.168.11.1&&udp.p

No.	Time	Source	Destination	Protocol	Length	
	0.000000000	192.168.11.105	192.168.11.1	DNS	86	Standard query 0x1 addr.arpa
	0.011995000	192.168.11.1	192.168.11.105	DNS	109	Standard query re

图 10-11　根据 IP+端口过滤

10.3.5　ngrep 过滤

ngrep 是一个基于 Libpcap 开发包设计的工具，可以根据数据包中的字符串、二进制数据识别出感兴趣的数据包，但它不具备数据流重组的功能。如果匹配的内容是跨数据包的，则 ngrep 无法检测，如果 ngrep 检测到匹配，则意味着该数据包的匹配，而不是数据流，过滤过程如图 10-12 所示。

```
VirtualUSMAllInOne:~# ngrep -I passwd.pcap "password"
input: passwd.pcap
match: password
################################################################################################
################################################################################################
################################################################################################
########################################################################################
T 211.150.65.169:80 -> 192.168.11.105:43945 [A]
  ="pass" type="password" style=" float:left;width:126px; height:30px; line-height:30px;border:none; backgroun
  d:url(images/index_94.jpg) no-repeat;">....            <a style="display:block; float:left; height:30px; lin
  e-height:30px; width:80px;" href="http://wmnew.263.net/setpass.html" target="_blank" class="wangji">........
  .......</a></div>..        <div align="center" style="margin-top:26px; width:310px; float:left; height:39px;
  ">..          <input name="submitnew" type="button" value="" onclick="login('new');" style="width:266px; hei
  ght:39px; border:none; background:url(images/index_32.jpg) no-repeat;cursor:pointer">..       </div>..
     <div align="center"   style="margin-top:8px;    width:310px; float:left; height:39px;">.... ..  <a href="
  http://mbill.263.net/frontuser/indexreg.jsp" target="_blank"><Img src="images/index_35.jpg" width="266" heig
  ht="40" style="cursor:pointer"></a> </div>......  ......  ......  ......  ......  ......  ......  ....</form
  >..  ......  ......  ..      </div>..    </div>..    .... ........    ..    ..    ..    ..    ....    <div
  class="taocan">  <a href="/web/more19_1.html"><img src="images/index_96.jpg" width="174" height="35"></a></d
  iv>..    <div class="taocan_c"> <a href="http://mbill.263.net/frontuser/indexreg.jsp"  target="_blank" class
  ="taocan_c_a1">.........-.........</a> <a href="http://mbill.263.net/frontuser/indexreg.jsp"  target="_blank
  "  class="
```

图 10-12　ngrep 过滤

以上实例如果需要用十六进制方式过滤可以使用“-x”参数，加入更多过滤条件的例子如下，仅针对所有源 IP 地址为 192.168.1.120 且目的端口为 80 的匹配数据包：

```
#ngrep -I passwd.pcap "pass" 'src host 192.168.1.120 and port 80'
```

10.4 命令行工具 tshark 和 dumpcap

上一节讲解了许多过滤方法，将这些方法组合起来应用就可发挥出强大的过滤效果。Wireshark 除了 GUI 工具外还有另外两个命令行工具——tshark、dumpcap，在远程环境下相当实用。

10.4.1 tshark 应用基础

下面简单介绍 tshark 常用的一些功能。Tshark 是命令行版的 Wireshark 流量分析器，依赖于 libpcap 流量捕获库来访问数据包，平常远程登录时抓包很好用，而且可以直接解析，但总是不记得如何打印出原本十六进制的数据，操作命令如下：

```
#tshark -w test.pcap -i eth0 -q
```

- -w：将抓包的数据写入文件中。
- -i：指定要抓包的接口名称。
- -q：安静，在远程时最有用，否则会抓到 SSH 的报文。
- -r：指定要读取的包文件。
- -x：将十六进制原始包数据打印出来。
- -V：显示详细信息。

操作举例：以秒为单位，统计 IP 地址 192.168.120.78 的封包、字节数量，操作如图 10-13 所示。

```
#tshark -z io,stat,1,ip.addr==192.168.120.78
```

```
alienvault:/var/ossim/traffic# tshark -z io,stat,1,ip.addr==192.168.120.78
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
  0.000000 192.168.120.78 -> 192.168.120.2 ICMP 98 Echo (ping) request  id=0x4eb3, seq=1/256, ttl=64
  0.000857 192.168.120.2 -> 192.168.120.78 ICMP 98 Echo (ping) reply    id=0x4eb3, seq=1/256, ttl=12
8
  4.999021 Vmware_bd:68:47 -> Vmware_fa:57:86 ARP 42 Who has 192.168.120.2?  Tell 192.168.120.78
  5.063192 Vmware_fa:57:86 -> Vmware_bd:68:47 ARP 60 192.168.120.2 is at 00:50:56:fa:57:86
^C4 packets captured

===================================
| IO Statistics                   |
|                                 |
| Interval size: 1 secs           |
| Col 1: ip.addr==192.168.120.78  |
|---------------------------------|
|          |1                |    |
| Interval
| Interval | Frames | Bytes |     |
|--------------------------|     |
| 0 <> 1 |      2 |   196 |     |
| 1 <> 2 |      0 | 34298784 |     |
| 2 <> 3 |      0 | 34298784 |     |
| 3 <> 4 |      0 | 34298784 |     |
| 4 <> 5 |      0 | 34298784 |     |
| 5 <> 5 |      0 | 34298784 |     |
===================================
alienvault:/var/ossim/traffic#
```

图 10-13　过滤出指定 IP 的数据包

实际上通过对网络数据包的分析，可以找到不少网络攻击过程中的奥秘，因为所有的网络攻击行为是通过数据包传送，而每种不同的攻击方式在网络上都会产生某种行为特征，比如说，当 Syn Flooding 攻击时，网络上立刻会产生大量带有 SYN 标志位的数据包，通过抓包就能发现，再如 Code red 蠕虫在感染 IIS 时，在网络上也会出现发送给 Web 服务器 80 端口的数据包，而且其中包含“GET /’ .* ‘.ida?’ * ‘XX’ * ‘%u’ {4} ‘%u780’ * ‘=’ .{7} ‘HTTP /1.0\r\n’”特征的字符串特征信息。

作为网络安全人员，我们要善于利用 Web 界面的 Wireshark 来分析网络的攻击行为。有的攻击产生小包（< 64 Bytes），而有的攻击则会产生巨人帧，我们通过不断总结攻击行为，将行为特征记录到数据库，这样就能形成一个适合企业自身的病毒查询知识库。

10.4.2 dumpcap 使用

dumpcap 使用和 tcpdump 类似。下面展示利用 dumpcap 捕获 ICMP 数据包的例子，如图 10-14 所示。

```
VirtualAllInOne6x1GB:/tmp# dumpcap -i eth0 -c 2 -w icmp.pcap -f "icmp and host 192.168.91.1"
Capturing on eth0
File: icmp.pcap
Packets captured: 2
Packets received/dropped on interface eth0: 0/0 (0.0%)
VirtualAllInOne6x1GB:/tmp# tcpdump -r icmp.pcap
reading from file icmp.pcap, link-type EN10MB (Ethernet)
22:39:38.729026 IP localhost > VirtualAllInOne6x1GB.alienvault: ICMP echo request, id 1, seq 5406, length 40
22:39:38.729057 IP VirtualAllInOne6x1GB.alienvault > localhost: ICMP echo reply, id 1, seq 5406, length 40
VirtualAllInOne6x1GB:/tmp# _
```

图 10-14 捕获 ICMP 协议

dumpcap 在 eth0 监听，将捕获 192.168.91.1 的 icmp 包，保存到当前目录下 icmp.pcap 文件中，然后我们用 tcpdump 工具来读取该包。这条命令的功能同样通过 tcpdump 也能实现，再用 dumpcap 命令实现，这样不重复吗？其实在 dumpcap 中，不必加入-s 参数指定包长度，dumpcap 默认抓包使用最大值。

10.4.3 用 tshark 分析 pcap

因为 dumpcap 默认抓包为最大值，以下试验的 pcap 文件均由 dumpcap 工具抓取，存储为 pcap 格式。

实例 1：读取数据包，操作如图 10-15 所示。

```
#tshark -r icmp.pcap
```

```
VirtualAllInOne6x1GB:/tmp# tshark  -r icmp.pcap
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
  1 0.000000000 192.168.91.1 -> 192.168.91.226 ICMP 74 Echo (ping) request  id=0x0001, seq=7979/11039, ttl=64
  2 0.000171000 192.168.91.226 -> 192.168.91.1 ICMP 74 Echo (ping) reply    id=0x0001, seq=7979/11039, ttl=64
VirtualAllInOne6x1GB:/tmp#
```

图 10-15 读取 pcap

实例 2：人性化方式显示时间。从输出看出 tshark 默认显示时间为 0，接着是第二个包逝去的时间，但这种时间显示方式难以看懂，我们换种方式显示，使用了“-t ad”开关，如图 10-16 所示，这样更方便。

```
VirtualAllInOne6x1GB:/tmp# tshark  -t ad -r icmp.pcap
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
  1 2015-07-09 23:22:21.509569000 192.168.91.1 -> 192.168.91.226 ICMP 74 Echo (ping) request  id=0x0001, seq=7979/11039, ttl=64
  2 2015-07-09 23:22:21.509740000 192.168.91.226 -> 192.168.91.1 ICMP 74 Echo (ping) reply    id=0x0001, seq=7979/11039, ttl=64
VirtualAllInOne6x1GB:/tmp# _
```

图 10-16　人性化显示时间

实例 3：在 tshark 中显示 ICMP 应答响应，如图 10-17 所示。

```
VirtualAllInOne6x1GB:/tmp# tshark -t ad -r icmp.pcap -R "icmp.type == 0"
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
  2 2015-07-09 23:22:21.509740000 192.168.91.226 -> 192.168.91.1 ICMP 74 Echo (ping) reply    id=0x0001, seq=7979/11039, ttl=64
VirtualAllInOne6x1GB:/tmp#
```

图 10-17　显示 ICMP 应答响应

实例 4：显示包详细信息。如果在 tshark –t ad –r icmp.pcap –R "icmp.type = = 0"这条命令后面加上-x -V 参数，分别表示以十六进制和详细解码，如图 10-18 所示。

```
VirtualAllInOne6x1GB:/tmp# tshark -t ad -r  icmp.pcap -R "icmp.type == 0" -x -V
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
    Interface id: 0
    WTAP_ENCAP: 1
    Arrival Time: Jul  9, 2015 23:22:21.509740000 EDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1436498541.509740000 seconds
    [Time delta from previous captured frame: 0.000171000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000171000 seconds]
    Frame Number: 2
    Frame Length: 74 bytes (592 bits)
    Capture Length: 74 bytes (592 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
Ethernet II, Src: Vmware_ca:2d:52 (00:0c:29:ca:2d:52), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
    Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
        Address: Vmware_c0:00:08 (00:50:56:c0:00:08)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Vmware_ca:2d:52 (00:0c:29:ca:2d:52)
        Address: Vmware_ca:2d:52 (00:0c:29:ca:2d:52)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.91.226 (192.168.91.226), Dst: 192.168.91.1 (192.168.91.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 60
    Identification: 0xee64 (61028)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x5428 [correct]
        [Good: True]
        [Bad: False]
    Source: 192.168.91.226 (192.168.91.226)
```

图 10-18　显示包的详细信息

在 SIEM 控制台中的这段代码/usr/share/ossim/www/forensics/base_paylad_tshark_tree.php，使用 tshark 来读取抓取 pcap 包，数据分析后最终通过 Web 界面展示给用户。更多抓包样本从这个链接获取：https://wiki.wireshark.org/SampleCaptures。

实例 5：搜索 SMTP 流量，这里使用了 smtp.req.command 过滤器，如图 10-19 所示。

```
VirtualAllInOne6x1GB:/tmp# tshark -t ad -r smtp.pcap -R 'smtp.req.command'
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
  7 2009-10-05 02:06:08.224809    10.10.1.4 -> 74.53.140.153 SMTP 63 C: EHLO GP
 10 2009-10-05 02:06:08.568729    10.10.1.4 -> 74.53.140.153 SMTP 66 C: AUTH LOGIN
 12 2009-10-05 02:06:08.911655    10.10.1.4 -> 74.53.140.153 SMTP 84 C: Z3VycGFydGFwQHBhdHJpb3RzLmlu
 14 2009-10-05 02:06:09.254118    10.10.1.4 -> 74.53.140.153 SMTP 72 C: cHVuamFiQDEyMw==
 16 2009-10-05 02:06:09.614414    10.10.1.4 -> 74.53.140.153 SMTP 90 C: MAIL FROM: <gurpartap@patriots.in>
 18 2009-10-05 02:06:09.957250    10.10.1.4 -> 74.53.140.153 SMTP 93 C: RCPT TO: <raj_deol2002in@yahoo.co.in
 20 2009-10-05 02:06:10.320203    10.10.1.4 -> 74.53.140.153 SMTP 60 C: DATA
 54 2009-10-05 02:06:14.763825    10.10.1.4 -> 74.53.140.153 SMTP 60 C: QUIT
```

图 10-19　搜寻 SMTP 流量

实例 6：搜寻两主机间通信。下面的例子需要过滤两条主机之间的流量，首先从我的博客下载 aim.pcap 包，接着用 tshark 提取从网络捕获的 aim.pcap 包中的所有 AIM 消息，输入如下命令：

```
#tshark -r aim.pcap -d tcp.port==443,aim -T fields -n -e
"aim.messageblock.message"
```

接着你会看到它们之间的通信。下面我们利用 tcpdumpde BPF 过滤功能，将主机之间的 SSL 通信提取出来，如图 10-20 所示。

```
VirtualAllInOne6x1GB:/tmp# tcpdump -s 0 -r aim.pcap -w aim-talker.pcap 'host 64.12.24.50 and host 192.168.1.158'
reading from file aim.pcap, link-type EN10MB (Ethernet)
VirtualAllInOne6x1GB:/tmp# tshark -r aim-talker.pcap
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
  1   0.000000 192.168.1.158 -> 64.12.24.50  SSL 60 Continuation Data
  2   0.000579  64.12.24.50 -> 192.168.1.158 TCP 60 https > 51128 [ACK] Seq=1 Ack=7 Win=64240 Len=0
  3  15.044068 192.168.1.158 -> 64.12.24.50  SSL 243 Continuation Data
  4  15.044588  64.12.24.50 -> 192.168.1.158 TCP 60 https > 51128 [ACK] Seq=1 Ack=196 Win=64240 Len=0
  5  15.135701 192.168.1.158 -> 64.12.24.50  SSL 94 Continuation Data
  6  15.135706  64.12.24.50 -> 192.168.1.158 TCP 60 https > 51128 [ACK] Seq=1 Ack=236 Win=64240 Len=0
  7  15.152349  64.12.24.50 -> 192.168.1.158 SSL 263 Continuation Data
```

图 10-20　两主机间 SSL 通信

用 tshark 抓取 HTTP 过滤请求操作如下：

```
alienvault:~# tshark 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)' -R 'http.re
quest.method == "GET" || http.request.method == "HEAD"'
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
  0.000000 192.168.11.129 -> 202.108.33.60 HTTP 287 GET / HTTP/1.0
  2.028938 192.168.11.129 -> 202.108.33.60 HTTP 290 GET / HTTP/1.0
 74.843823 192.168.11.129 -> 66.102.251.33 HTTP 283 GET / HTTP/1.0
 77.072560 192.168.11.129 -> 202.108.33.60 HTTP 287 GET / HTTP/1.0
 79.094601 192.168.11.129 -> 202.108.33.60 HTTP 290 GET / HTTP/1.0
 87.197244 192.168.11.129 -> 129.42.38.1  HTTP 282 GET / HTTP/1.0
 89.532045 192.168.11.129 -> 221.204.163.25 HTTP 286 GET / HTTP/1.0
^Z
```

同样，HTTP 中与服务器交互的其他方法 POST、PUT、DELETE 都可以根据以上命令轻松实现。下面再看一个例子，过滤 HTTP 请求中的网址，操作如下：

```
alienvault:~# tshark -i eth0 tcp port 80 -R 'ip' -T fields -e ip.src -R 'http.request' -T fields -e http.host
-e http.request.uri
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
192.168.11.129  ibm.com /
192.168.11.129  www.ibm.com     /
192.168.11.129  www.ibm.com     /us/en/
192.168.11.129  mail.126.net    /
192.168.11.129  www.sina.com.cn /
192.168.11.129  tech.sina.com.cn        /z/sinawap/
192.168.11.129  weibo.com       /58330
192.168.11.129  passport.weibo.com      /visitor/visitor?entry=miniblog&a=enter&url=http%3A%2F%2Fweibo.com%2F5
8330&domain=.weibo.com&sudaref=http%3A%2F%2Ftech.sina.com.cn%2Fz%2Fsinawap%2F&ua=php-sso_sdk_client-0.6.14&_ra
nd=1442011777.0789
192.168.11.129  www.google.com  /
```

10.5 使用 tcpdump 过滤器

平时我们只用 tcpdump 进行抓包分析，进行数据包分析主要掌握协议字段（从协议字段中提取数据）、匹配模式（通过查找数据包中的特定值找出感兴趣数据包），过滤器是一种高级用法，用于限制 tcpdump 及其他工具显示或捕获的流量。过滤器有时被简称为 BPF（Berkeley Packet Filter），过滤就是去除冗余的部分，标记出感兴趣内容，下面举几个操作实例。

10.5.1 tcpdump 过滤器基础

我们首先了解 tcpdump 的几个重要参数。

- -n：告知 tcpdump 不要通过 DNS 查询将 IP 地址解析为主机名。
- -s: 告知每个数据包捕获多少字节， -s 0 表示整个数据包。
- -c: 捕获多少个数据包。

实例 1：只捕获 10 个 ICMP 数据包，操作如下：

```
VirtualUSMAllInOne:~# tcpdump -n -i eth0 -c 10 -w icmp.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10 packets captured
10 packets received by filter
0 packets dropped by kernel
VirtualUSMAllInOne:~#
```

为了读取跟踪，使用 tcpdump 的-r 参数：

```
#tcpdump -n -r icmp.pcap
```

```
VirtualUSMAllInOne:~# tcpdump -n -r icmp.pcap
reading from file icmp.pcap, link-type EN10MB (Ethernet)
11:44:41.076050 IP 192.168.11.6 > 192.168.11.105: ICMP echo request, id 1, seq 769, length 40
11:44:41.076143 IP 192.168.11.105 > 192.168.11.6: ICMP echo reply, id 1, seq 769, length 40
11:44:41.078034 IP 192.168.11.6 > 61.148.185.181: ICMP echo request, id 1, seq 770, length 40
11:44:41.080897 IP 61.148.185.181 > 192.168.11.6: ICMP echo reply, id 1, seq 770, length 40
11:44:41.214701 IP 192.168.11.1.4680 > 192.168.11.6.21: Flags [S], seq 3135961814, win 5840, options [mss 1460,sa
ckOK.TS val 300313904 ecr 0.nop.wscale 11. length 0
```

实例 2：查找特定端口流量。

如果不使用 icmp，还可以通过 TCP、UDP 选项来捕获其他特定流量，例如为了查找 DNS 故障，可捕获端口 53 的数据包。

```
VirtualUSMAllInOne:~# tcpdump -n -i eth0 -s 0 port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:50:18.594789 IP 192.168.11.6.59388 > 192.168.11.1.53: 29920+ A? clients4.google.com. (37)
11:50:18.601524 IP 192.168.11.1.53 > 192.168.11.6.59388: 29920 8/0/0 CNAME clients.l.google.com., CNAME clients-c
hina.l.google.com., A 74.125.204.113, A 74.125.204.138, A 74.125.204.139, A 74.125.204.100, A 74.125.204.101, A 7
4.125.204.102 (185)
11:50:20.951587 IP 192.168.11.105.51932 > 192.48.79.30.53: 37478 [1au] A? www.ibm.com. (40)
11:50:21.095875 IP 192.48.79.30.53 > 192.168.11.105.51932: 37478- 0/12/11 (875)
11:50:21.097682 IP 192.168.11.105.31164 > 184.26.161.64.53: 29691 [1au] A? www.ibm.com. (40)
11:50:21.097867 IP 192.168.11.105.38222 > 72.246.46.67.53: 62855% [1au] A? usc2.akam.net. (42)
11:50:21.098047 IP 192.168.11.105.59853 > 95.100.174.67.53: 39529% [1au] AAAA? usc2.akam.net. (42)
11:50:21.098232 IP 192.168.11.105.2074 > 23.211.61.67.53: 9482% [1au] A? ns1-206.akam.net. (45)
11:50:21.098490 IP 192.168.11.105.17588 > 193.108.91.67.53: 11650% [1au] AAAA? ns1-99.akam.net. (44)
11:50:21.098730 IP 192.168.11.105.19499 > 96.7.49.67.53: 51016% [1au] AAAA? ns1-206.akam.net. (45)
```

实例 3：如果只为了捕获 80 端口上的 TCP 流量，则添加 and tcp 参数即可。

```
VirtualUSMAllInOne:~# tcpdump -n -c 2 -i eth0 -s 0 port 80 and tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:53:52.485394 IP 192.168.11.105.60173 > 74.121.134.252.80: Flags [F.], seq 2727679738, ack 3323544625, win 85
, length 0
11:53:52.485885 IP 192.168.11.105.49690 > 222.161.252.52.80: Flags [P.], seq 628303813:628304187, ack 109354124
 win 74, length 374
2 packets captured
2 packets received by filter
0 packets dropped by kernel
VirtualUSMAllInOne:~#
```

10.5.2 其他常见过滤器使用方法

在上面实例中，icmp.pcap 中过滤某台主机流量：

```
#tcpdump -n -r icmp.pcap host 192.168.11.1
```

来自（源地址过滤）某主机的流量：

```
#tcpdump -n -r icmp.pcap src host 192.168.11.1
```

过滤捕获流向某主机的流量：

```
#tcpdump -n -r icmp.pcap dst host 192.168.11.1
```

捕获流向某网络的流量：

```
#tcpdump -n -r icmp.pcap dst net 192.168.11.0
```

有时我们在检查网络连通情况时，需要过滤出 ICMP 的应答响应，操作如下：

```
VirtualUSMAllInOne:~# tcpdump -n -r icmp.pcap 'icmp[icmptype]=icmp-echoreply' and dst host 192.168.11.6
reading from file icmp.pcap, link-type EN10MB (Ethernet)
11:44:41.076143 IP 192.168.11.105 > 192.168.11.6: ICMP echo reply, id 1, seq 769, length 40
11:44:41.080897 IP 61.148.185.181 > 192.168.11.6: ICMP echo reply, id 1, seq 770, length 40
VirtualUSMAllInOne:~#
```

使用 tcpdump 除了可以更详细地显示流量，还能从结果中提取更多细节，例如，加入-tttt 参数告诉 tcpdump 将时间戳显示为年月日的格式，-e 参数添加 2 层头部信息，-XX 参数以十六进制和 ASCII 格式显示所有头部和数据，如图 10-21 所示。

```
alienvault:/tmp# tcpdump -n -tttt -e  -XX -r /tmp/icmp.pcap  'icmp[icmptype]=icm
p-echoreply' and dst host 192.168.91.1
reading from file /tmp/icmp.pcap, link-type EN10MB (Ethernet)
2015-07-08 03:40:26.821279 00:0c:29:94:d8:49 > 00:50:56:c0:00:08, ethertype IPv4
 (0x0800), length 74: 192.168.91.227 > 192.168.91.1: ICMP echo reply, id 1, seq
31, length 40
        0x0000:  0050 56c0 0008 000c 2994 d849 0800 4500  .PV.....)..I..E.
        0x0010:  003c 1183 0000 4001 3109 c0a8 5be3 c0a8  .<....@.1...[...
        0x0020:  5b01 0000 553c 0001 001f 6162 6364 6566  [...U<....abcdef
        0x0030:  6768 696a 6b6c 6d6e 6f70 7172 7374 7576  ghijklmnopqrstuv
        0x0040:  7761 6263 6465 6667 6869                 wabcdefghi
```

图 10-21　显示 ICMP 应答响应

下面这个例子用来研究多个抓包内容，在/tmp 目录下具有多个抓包文件 icmp.pcap、ip.pcap 等。当你想搜索所有 pcap 文件时该怎么操作？下面我们利用 for 循环和 find 命令可在所有文件中查找主机 192.168.91.227 的 TCP 流量信息，如图 10-22 所示。

```
alienvault:/tmp# for i in `find /tmp/ -type f `; do tcpdump -n -c 1 -r $i host 1
92.168.91.227 and tcp; done
tcpdump: truncated dump file; tried to read 4 file header bytes, only got 0
tcpdump: unknown file format
tcpdump: unknown file format
reading from file /tmp/icmp.pcap, link-type EN10MB (Ethernet)
reading from file /tmp/wireshark_eth0_20150708032756_8NwPGv, link-type EN10MB (E
thernet)
tcpdump: truncated dump file; tried to read 4 file header bytes, only got 0
reading from file /tmp/ip.pcap, link-type EN10MB (Ethernet)
03:54:59.890861 IP 216.58.221.68.80 > 192.168.91.227.42371: Flags [R.], seq 9416
33309, ack 4107026272, win 64240, length 0
reading from file /tmp/123.pcap, link-type EN10MB (Ethernet)
03:26:11.060066 IP 192.168.91.227.60675 > 64.62.160.45.80: Flags [S], seq 230669
3241, win 5840, options [mss 1460,nop,nop,sackOK,nop,wscale 10], length 0
alienvault:/tmp# _
```

图 10-22　过滤出多个包的内容

10.5.3　通过 Traffic Capture 抓包存储

从维护的角度看，对于抓包存放的位置是操作者必须了解的内容，系统抓的 pcap 数据包会存放在/var/ossim/traffic/目录下。例如：Netscan_admin_138901088_10_192.168.150.130.pcap。如果是分布式 OSSIM，则抓的包存放在相应 Sensor 的/var/ossim/traffic/目录中。

10.6 针对 IE 浏览器漏洞的攻击分析

下面将要为读者做一个漏洞的攻击分析，在 2010 年有一个 0day 漏洞名叫极光漏洞，它是 IE 的一个缓冲区溢出漏洞，利用这个漏洞进行攻击的程序实例介绍如下。

受害者 IP：192.168.100.206，攻击者 IP：192.168.100.200。

由于在 Windows 系统的 IE6/7/8 等版本中，存在零日漏洞，没有修复此漏洞的客户端，有可能被攻击者的网页挂马导致任意代码执行，这里所举的例子中，受害者就是使用了 Windows XP+SP3+IE6 的系统。下面讲述通过 OSSIM 抓包并分析可疑数据包的过程。

在 OSSIM 下抓包发现了一些可疑数据包，我们分析这些数据包的内容。首先受害者和攻击者之间进行了三次握手，而且初始化连接的目标端口是 80，很显然属于 HTTP 流量。从第 13 个数据包，能看出这是一个对/info 的 HTTP GET 请求，在图 10-23 所示中用绿色标出的部分。

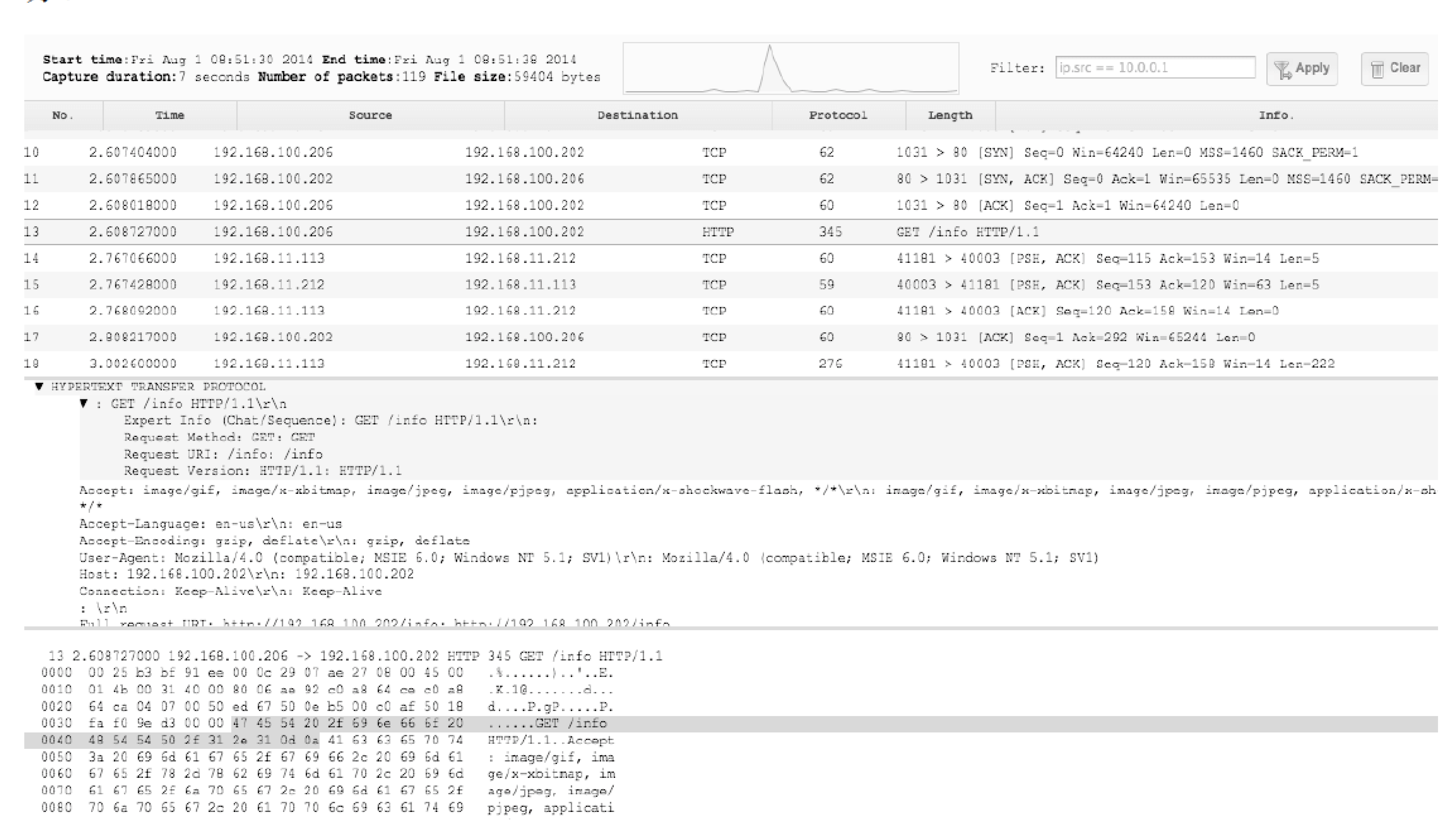


图 10-23　GET 请求

当攻击者的机器收到 GET 请求后，并在第 50 个数据包中返回了一个 302 返回码，其含义是用于浏览器重定向到另一个页面，如图 10-24 所示。

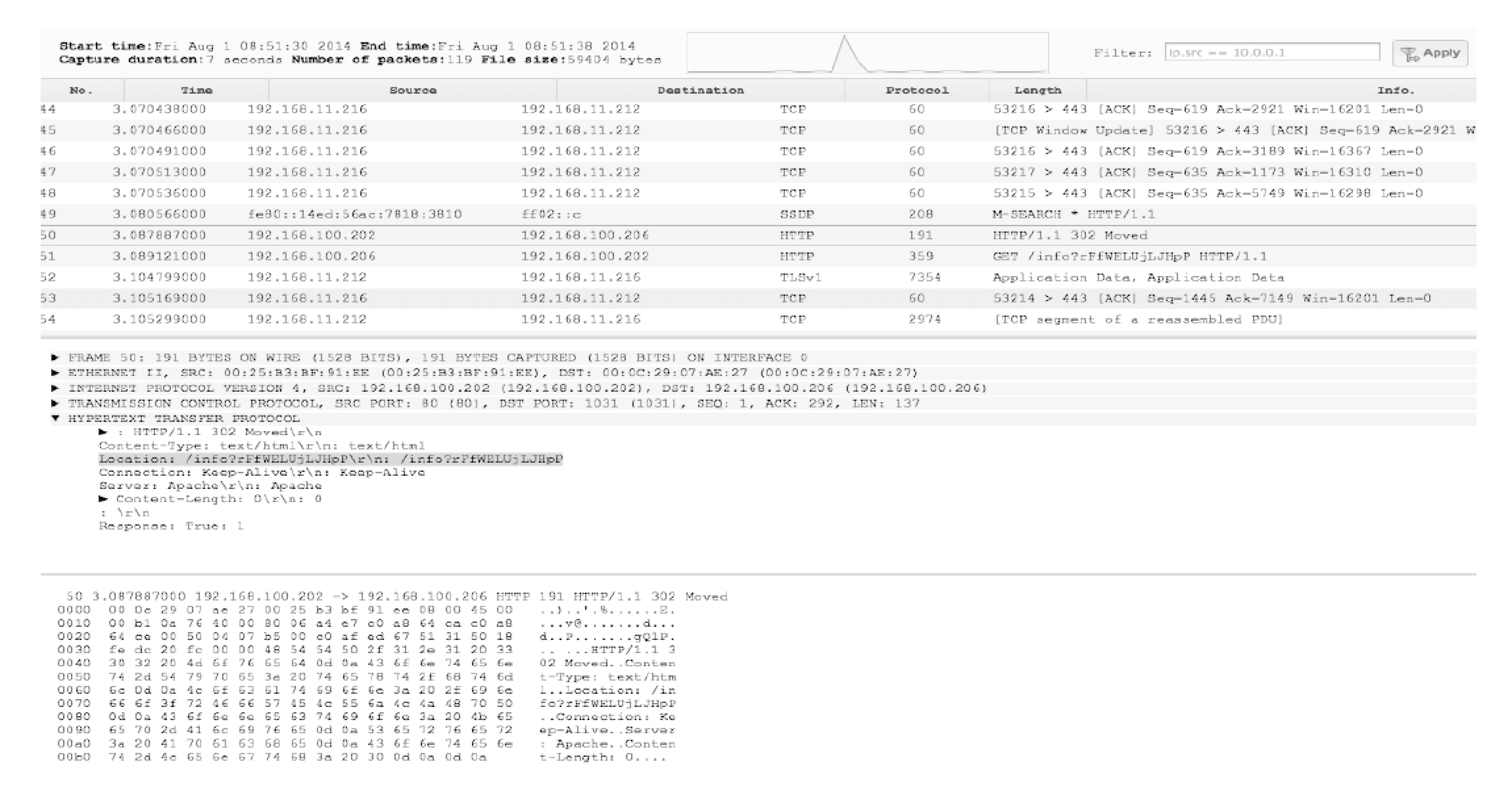


图 10-24　浏览器被重定向

注意这个 Location 域，它指明重定向的位置是/info?rFfWELUjLJHpP，这些内容好像是<script>标签内的一些随机字母，而这些类似乱码的随机字母，则可能被攻击者经过了特殊编码而逃避检测。接下来，我们发现了一些更有用的信息，如图 10-25 所示。

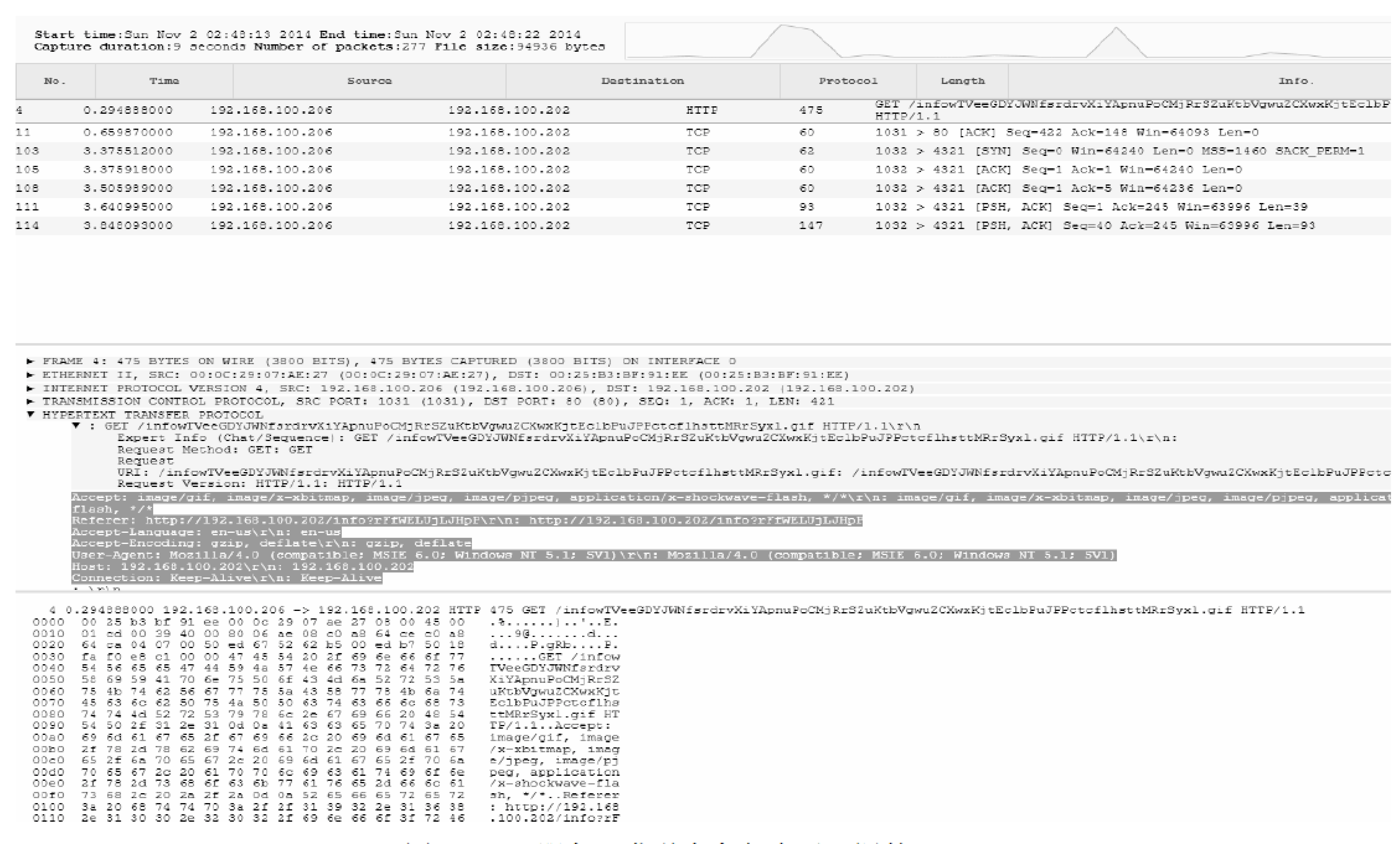


图 10-25　服务器发送内容包含了可疑的 iframe

注意图中给出的 URI:

```
/infowTVeeGDYJWNfsrdrvXiYApnuPoCMjRrSZuKtbVgwuZCXwxKjtEclbPuJPPctcflhsttMRrSyxl.gif:
```

这是个恶意的嵌套，很可能是包含在标签内，这些文本很长，包含了不可阅读的字符串，这是攻击者惯用的手法。这样用户无法察觉，这个奇怪的 GIF 文件有什么用呢，这绝不是什么图片。另外，大家注意 Graphs 按钮内容，显示了 Wireshark 的 I/O 图像，它可以描绘网络上的吞吐量，分析者利用这些图便可找到不同协议数据吞吐的峰值，如图 10-26 所示。

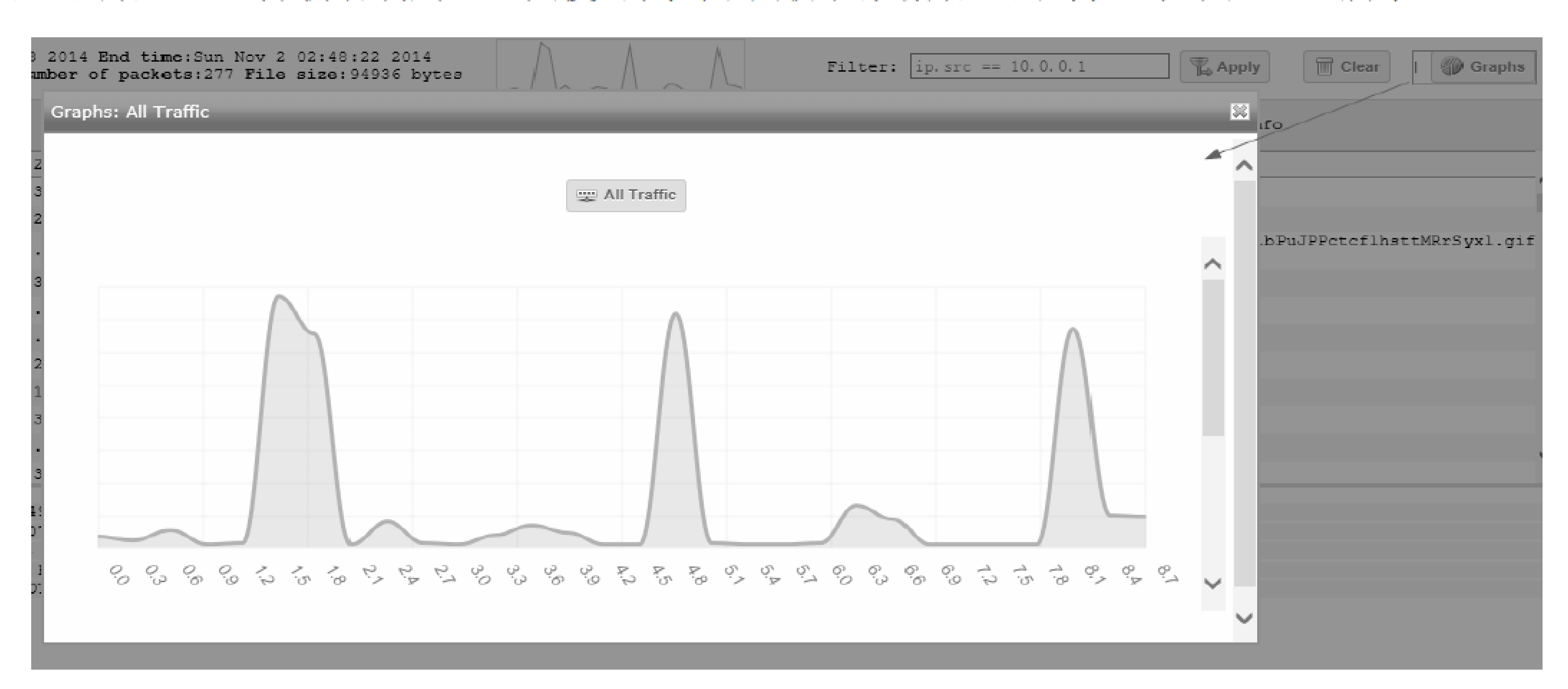


图 10-26　反映吞吐量变化

该 GIF 文件很有可能用来触发某种代码，如图 10-27 所示。

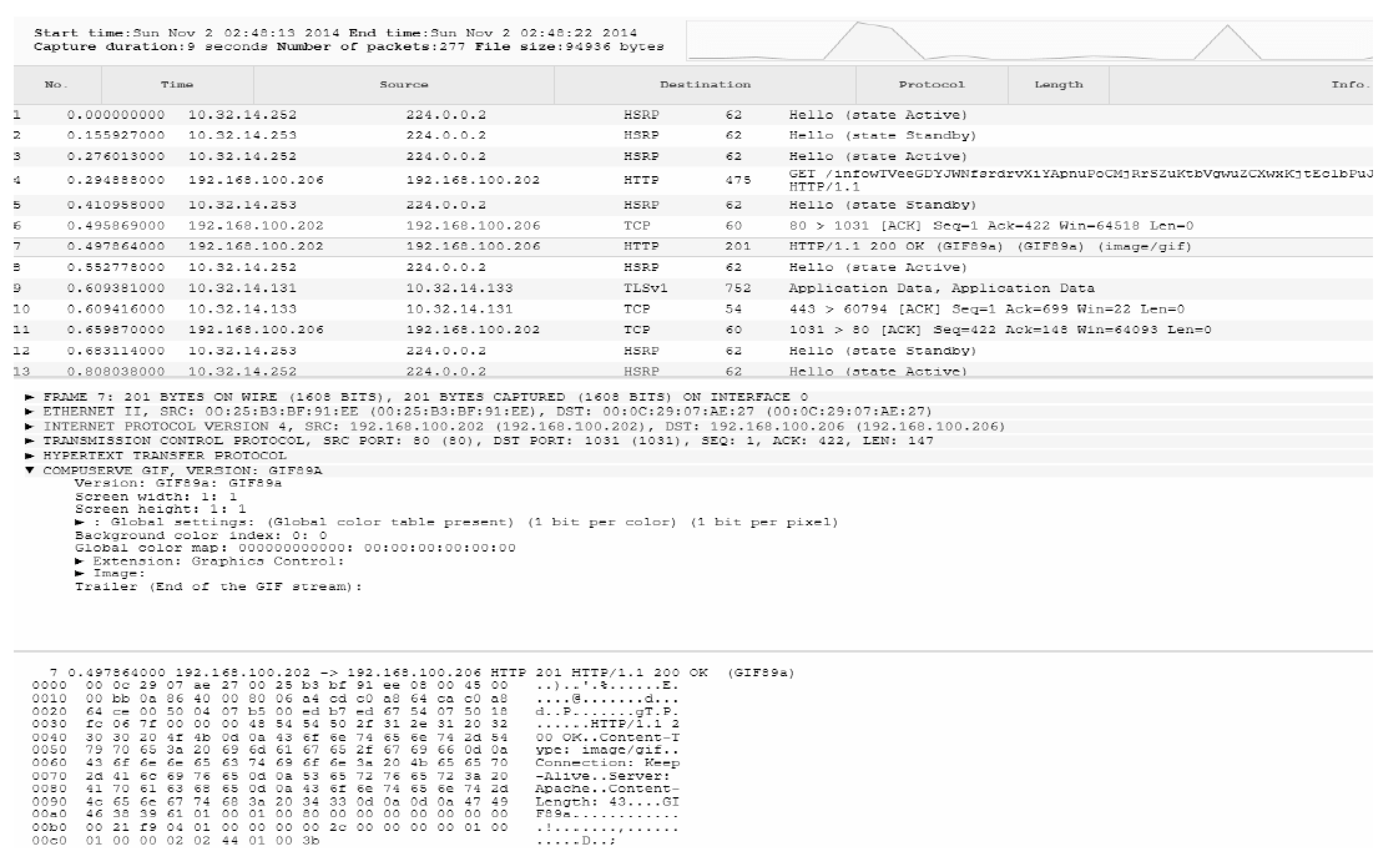


图 10-27　GIF 文件很可能用来被触发某种代码

从图 10-28 我们清楚看到一个 Windows 命令行解释器，这个 shell 是由受害者发给服务器的，这表明攻击者成功利用了漏洞，我们甚至可以看到攻击者和受害者的交互，即输入 dir 命令，列出受害者机器的目录。这样攻击者对受害者机器就有了管理权限。

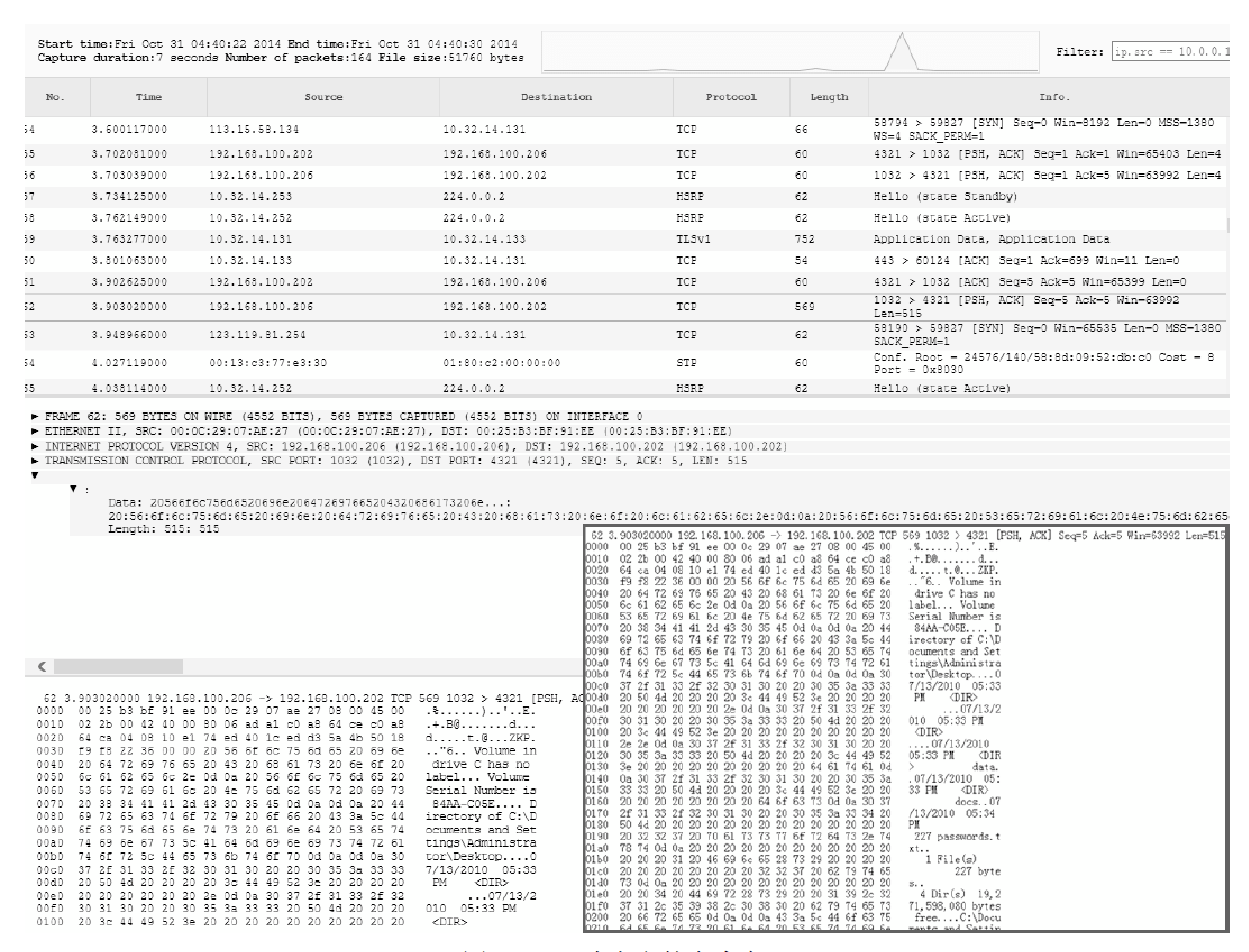


图 10-28　攻击者的命令交互

以上整个过程只需要几秒钟，而对于管理员很难捕获到这一变化。下面我们开始梳理 OSSIM 中的流量抓包功能（Traffic Capture），看看它如何找到这些可疑数据包。

首先，OSSIM 的一块网卡和交换机 SPAN 口连接，正常情况下能分析所有的数据包。某时刻遇到这样的场景，受害者接收到了可疑邮件里面的图片，并单击了该图片，这对于浏览者而言在平常不过了，可是图片背后的链接却向攻击者的恶意网站发送了一个 GET 请求。接着，攻击者的 Web 服务器向受害者发送了 302 重定向，这样一来，受害者的浏览器就向重定向后的 URL 发起了另一个 GET 请求。攻击者的 Web 服务器向这位客户端发送了含有恶意嵌套 GIF 图像连接的帧（iframe），受害者从服务器上下载了 GIF，这样利用 IE 浏览器漏洞，便执行了隐藏的 PayLoad，并打通了受害者到攻击者的网络通道，该 Payload 产生了一个命令行解释器到攻击者的计算机，以便远程控制目标。

应对这种攻击，首先还是需要在受害者计算机上安装最新系统并及时打系统补丁，对于网络管理方，利用前面介绍的 Snort 章节的内容，使用这个捕获文件为 IDS 创建一个签名，主要

是获取特征码，为所有包含 302 重定向到特定 URL 的 HTTP 流量也写一个签名，便有助于检测这种攻击。当然在生产环境中遇到的情况千差万别，用这一思路再加上一些耐心的调整，一定能够对你有所帮助。OSSIM 系统的漏洞扫描功能也能助一臂之力，如图 10-29 所示，漏洞库中就有极光 IE 0 day 漏洞的信息。

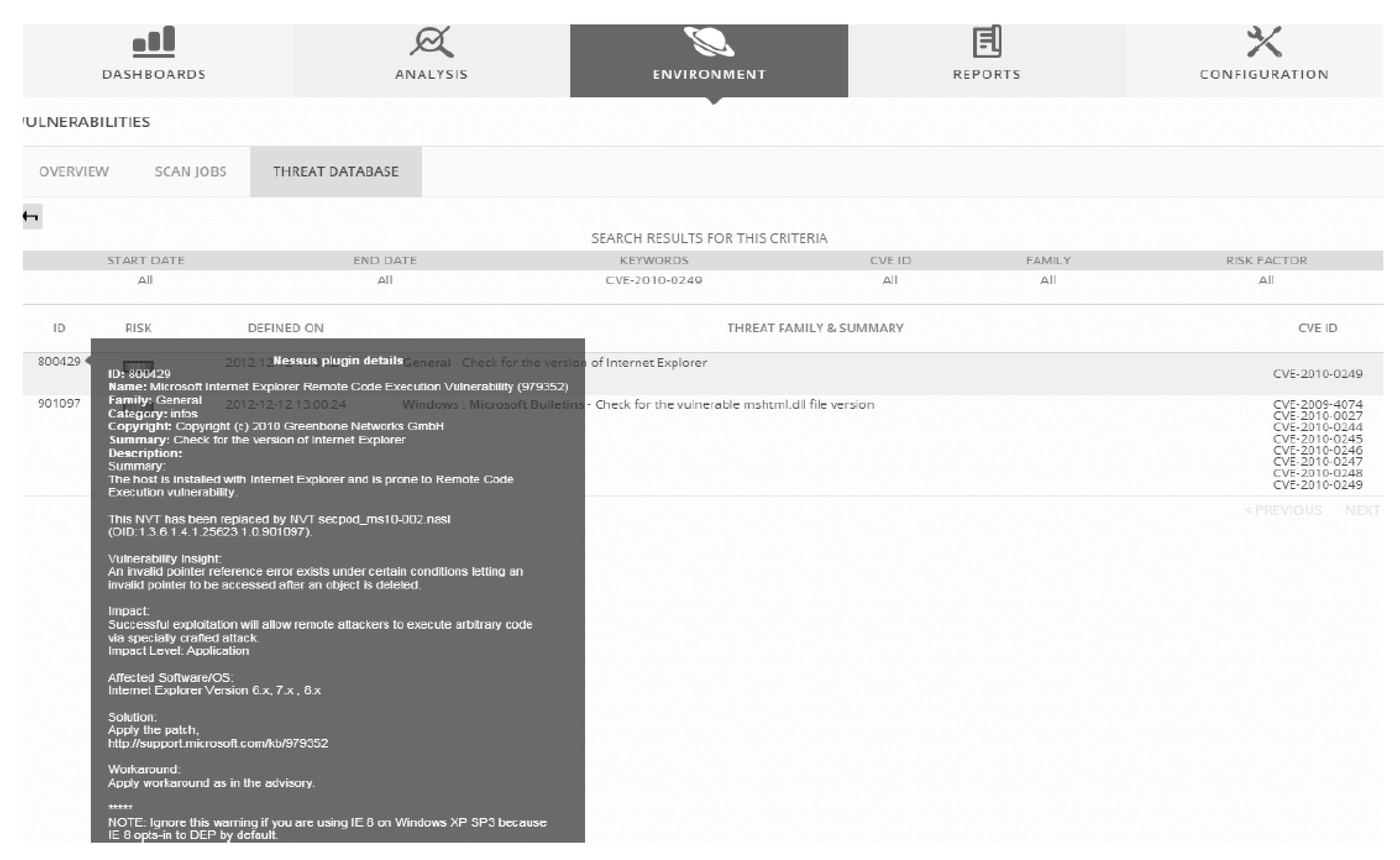


图 10-29　极光漏洞库查询

10.7 小结

本章主要讲解了包分析和过滤技术的诸多细节和技巧，主要讲解了 OSSIM 下基于 Web 方式的抓包技术，后台其实是 tshark 这个命令行工具在起作用，包括 tcpdump 的过滤技巧，并对各种端口协议和过滤方法做了详细讲解。数据包分析有很多乐趣，读者只有不断在实践中多加练习，灵活使用这些技巧，才能快速解决各种问题。

参考文献

1. 李晨光. Linux 企业应用案例精解. 第 2 版. 北京：清华大学出版社，2014
2. 李晨光. Unix/Linux 网络日志分析与流量监控. 北京：机械工业出版社，2015
3. OSSIM. http://www.alienvault.com/
4. Debian. https://www.debian.org/doc/debian-policy/
5. CVE. http://cve.mitre.org/
6. Snort. https://www.snort.org/faq
7. MySQL. http://dev.mysql.com/doc/
8. Rsyslog. http://www.rsyslog.com/faq/
9. Openvas. http://www.openvas.org/documentation.html